MPC5744P Reference Manual

Supports MPC5744P, MPC5743P, MPC5742P and MPC5741P

Document Number: MPC5744PRM
Rev. 6, 06/2016

Contents

Section number **Title** **Page**

Chapter 1
Preface

Chapter 2
Introduction

Chapter 3

Section number Title Page

Embedded Memories

Chapter 4
Signal Description

Chapter 5
Memory Map

Chapter 6

Functional Safety

Chapter 7
Chip Configuration

Section number Title Page

Chapter 8
Reset

Chapter 9
Device Configuration Format (DCF) Records

Chapter 10
Device Security

Chapter 11
Debug

Chapter 12
Power Management

Chapter 13
Clocking

Section number Title Page

Chapter 14
e200z4d Core Complex Overview

Chapter 15
Core Detailed Description

Chapter 16
System Integration Unit Lite2 (SIUL2)

Chapter 17
Crossbar Switch (XBAR)

Chapter 18
Crossbar Integrity Checker (XBIC)

Chapter 19
Peripheral Bridge (AIPS-Lite)

Chapter 20
System Memory Protection Unit (SMPU)

Chapter 21
Interrupt Controller (INTC)

Chapter 22
Enhanced Direct Memory Access (eDMA)

Section number **Title** **Page**

Chapter 23
Direct Memory Access Multiplexer (DMAMUX)

Chapter 24
Error Injection Module (EIM)

Chapter 25
Dual PLL Digital Interface (PLLDIG)

Chapter 26
Clock Monitor Unit (CMU)

Chapter 27
Clock Generation Module (MC_CGM)

Chapter 28
OSC Digital Interface (XOSC)

Chapter 29
IRCOSC Digital Interface

Chapter 30
RAM Controller (PRAMC)

Chapter 31
Flash Memory Controller (PFLASH)

Section number	Title	Page

Chapter 32
Embedded Flash Memory (c55fmc)

Chapter 33
Flash OTP Control

Section number Title Page

Chapter 34
Decorated Storage Memory Controller (DSMC)

Chapter 35
ADC Configuration

Chapter 36
Analog-to-Digital Converter (ADC)

Section number Title Page

Chapter 37
Temperature Sensor (TSENS)

Chapter 38
Sine Wave Generator (SGEN)

Chapter 39
Enhanced Motor Control Timer (eTimer)

Chapter 40
Motor Control Pulse Width Modulator Module (FlexPWM)

Section number Title Page

Chapter 41
Cross-Triggering Unit (CTU)

Section number Title Page

Chapter 42
System Timer Module (STM)

Chapter 43
Software Watchdog Timer (SWT)

Chapter 44
Periodic Interrupt Timer (PIT)

Chapter 45
CAN (FlexCAN)

Chapter 46
Zipwire

Chapter 47
Serial Interprocessor Interface (SIPI)

Chapter 48
LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

Chapter 49
Serial Peripheral Interface (SPI)

Chapter 50
FlexRay Communication Controller (FlexRay)

Section number Title Page

Chapter 51
SENT Receiver (SRX)

Chapter 52
LINFlexD

Section number | Title | Page

Chapter 53
10/100-Mbps Ethernet MAC (ENET)

Chapter 54
Reset Generation Module (MC_RGM)

Chapter 55
Boot Assist Module (BAM)

Chapter 56
System Status and Configuration Module (SSCM)

Chapter 57
Power Management Controller block (PMC)

Chapter 58
Power Control Unit (MC_PCU)

Chapter 59
Mode Entry Module (MC_ME)

Chapter 60
Core Debug Support

Section number | Title | Page

Chapter 61
JTAG Controller (JTAGC)

Section number | Title | Page

Chapter 62
Nexus Module

Section number	Title	Page

Chapter 63
Nexus Crossbar Multi-Master Client (NXMC)

Chapter 64
Nexus Port Controller (NPC)

Chapter 65

Nexus Aurora Link (NAL)

Chapter 66
Nexus Aurora PHY (NAP)

Chapter 67
Cyclic Redundancy Check (CRC) Unit

Chapter 68
Memory Error Management Unit (MEMU)

Chapter 69
Fault Collection and Control Unit (FCCU)

Chapter 70
Self-Test Control Unit (STCU2)

Chapter 71
Register Protection (REG_PROT)

Chapter 72
Wakeup Unit (WKPU)

Chapter 1
Preface

1.1 Overview

The primary objective of this document is to define the functionality of the MPC5744P microcontroller for use by software and hardware developers. The MPC5744P Qorivva microcontroller is based on the Power Architecture® developed by Freescale. The MPC5744P is a SafeAssure solution.

As with any technical documentation, it is the reader's responsibility to ensure he or she is using the most recent version of this document.

To locate any published errata or updates for this document, visit the Freescale Web site at http://www.freescale.com.

1.2 Device versions

This release of this document describes the functionality and programming model of the device intended for production.

The device has these part numbers: MPC5744P, MPC5743P, MPC5742P, and MPC5741P. The parts differ with regard to flash memory and system RAM sizes. This document uses MPC5744P as the generic part number.

1.3 Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MPC5744P device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

1.4 Document organization

This document includes two major sets of chapters:

- Chapters in the first set describe the device as a whole or device-specific information.
- Chapters in the second set describe the functionality of the individual modules on the device. These chapters are organized into the following groups:
    - Core and system modules
    - Clocking modules
    - Memories and memory interfaces
    - Motor control modules
    - Timers
    - Communication interfaces
    - Reset and boot modules
    - Power management modules
    - Debug
    - Safety modules
    - Other modules

1.5 Conventions

1.5.1 Numbering systems

The following suffixes identify different numbering systems:

This suffix	Identifies a
b	Binary number. For example, the binary equivalent of the number 5 is written 101b. In some cases, binary numbers are shown with the prefix *0b*.
d	Decimal number. Decimal numbers are followed by this suffix only when the possibility of confusion exists. In general, decimal numbers are shown without a suffix.
h	Hexadecimal number. For example, the hexadecimal equivalent of the number 60 is written 3Ch. In some cases, hexadecimal numbers are shown with the prefix *0x*.

1.5.2 Typographic notation

The following typographic notation is used throughout this document:

Example	Description
placeholder, x	Items in italics are placeholders for information that you provide. Italicized text is also used for the titles of publications and for emphasis. Plain lowercase letters are also used as placeholders for single letters and numbers.
`code`	Fixed-width type indicates text that must be typed exactly as shown. It is used for instruction mnemonics, directives, symbols, subcommands, parameters, and operators. Fixed-width type is also used for example code. Instruction mnemonics and directives in text and tables are shown in all caps; for example, BSR.
SR[SCM]	A mnemonic in brackets represents a named field in a register. This example refers to the Scaling Mode (SCM) field in the Status Register (SR).
REVNO[6:4], XAD[7:0]	Numbers in brackets and separated by a colon represent either: • A subset of a register's named field For example, REVNO[6:4] refers to bits 6–4 that are part of the COREREV field that occupies bits 6–0 of the REVNO register. • A continuous range of individual signals of a bus For example, XAD[7:0] refers to signals 7–0 of the XAD bus.

1.5.3 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
deasserted	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1). In some cases, deasserted signals are described as *negated*.
reserved	Refers to a memory space, register, field, or programming setting. Writes to a reserved location can result in unpredictable functionality or behavior. • Do not modify the default value of a reserved programming setting, such as the reset value of a reserved register field. • Consider undefined locations in memory to be reserved.
w1c	Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared."

1.6 Acronyms and abbreviations

Term	Meaning
AUTOSAR	Automotive Open System Architecture
GPIO	General-purpose I/O

Table continues on the next page...

Term	Meaning
IEEE	Institute for Electrical and Electronics Engineers
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
Mux	Multiplex
Rx	Receive
RTL	Register transfer language
SM	Safety Manual
TBD	To be determined
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

1.7 References

In addition to this reference manual, the following documents provide additional information on the operation of the MPC5744P:

- IEEE-ISTO 5001-2003 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture
- Power Architecture Book E V1.0 (http://www.freescale.com/files/32bit/doc/user_guide/BOOK_EUM.pdf)

Chapter 2
Introduction

2.1 Overview

The MPC5744P Qorivva microcontroller is based on the Power Architecture® developed by Freescale Semiconductor. It targets chassis and safety applications and other applications requiring a high Automotive Safety Integrity Level (SIL). The MPC5744P is a SafeAssure solution.

2.1.1 Overall architecture

All devices in this family are built around a safety concept based on delayed lock step, targeting an ISO26262 ASIL-D (Design) integrity level. According to FMEDA analysis, critical components of the microcontroller that must be replicated are the CPU core and DMA controller. Lock step Checking Units are implemented at each output of these blocks to compare the values between the redundant blocks.

In addition, the MPC5744P provides:

- A programmable Fault Collection and Control Unit (FCCU) to monitor the integrity status of the device and provide flexible safe state control.
- End-to-End Error Correcting Code (e2eECC) for improved fault tolerance and detection:
  - All bus masters generate Single Error Correcting and Double Error Detecting (SECDED) code for every bus transaction
  - SECDED covers 64-bit data and 29-bit address
  - ECC is stored in memories on write operations and validated by the master on every read operation

2.1.2 Core features

The host processor core of the MPC5744P is a CPU from the e200 family of compatible Power Architecture cores. The z425n3 dual issue core provides very high efficiency—high performance with minimum power dissipation—and operates at a maximum frequency of 200MHz.

The processor family implements low-cost versions of the *PowerISA 2.06* architecture. The z425n3 is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general purpose registers (GPRs).

An Embedded Floating-point (EFPU2) Auxiliary Processing Unit (APU) is provided to support real-time single-precision embedded numeric operations using the general-purpose registers.

A Lightweight Signal Processing Extension (LSP) APU is provided to support real-time SIMD fixed-point embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the GPRs.

The z425n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is documented in *PowerISA 2.06*, a separate document. Note that the base PowerISA 2.06 fixed-length 32-bit instruction set is not directly supported.

The z425n3 processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The z425n3 also contains an 8 KB Instruction Cache and a 4 KB Data Cache as well as a Nexus Class 3+ debug module.

A Memory Protection Unit (MPU) is also included which supports protections of various instruction and data memory areas.

The MPC5744P has two e200z4 CPU core complexes running in delayed lock step.

2.1.3 Memory hierarchy

The MPC5744P has a single level of memory hierarchy consisting of 384 KB on-chip SRAM and 64 KB tightly coupled local data SRAM with ECC and 2.5 MB on-chip flash memory including ECC. Both the SRAM and the flash memory can hold instructions and data.

2.1.4 Off-chip communication

Off-chip communication is performed by a suite of serial protocols including FlexRay, CANs, enhanced SPIs (DSPIs), SCIs (LINFlexD), SIPI (via LFAST), Ethernet, and the SENT sensor interface.

2.1.5 I/O control

The System Integration Unit “Lite2” (SIUL2) performs several chip-wide configuration functions. SIUL2 is not software compatible to the SIUL used on Freescale C90 products.

The SIUL2 controls pad configuration and general-purpose input/output (GPIO). External interrupts and reset control are also found in the SIUL2. The internal multiplexer sub-block (IOMUX) provides multiplexing options for the device pins’ output and input paths (for example, daisy chaining the DSPIs and external interrupt signal).

2.1.6 Peripheral set compatibility

The MPC5744P peripheral set is compatible with the MPC5643L, providing high-end electrical motor control capability with very low CPU intervention due to the on-chip Cross Triggering Unit (CTU). For more information, see Compatibility with MPC5643L.

2.1.7 Technology

The MPC5744P is developed with 55 nm embedded flash memory technology, providing a significant performance improvement.

2.2 Target applications

As a SafeAssure solution, the MPC5744P microcontroller targets applications requiring a high Automotive Safety Integrity Level (ASIL), especially:

- Chassis applications
- Electrical Stability Control (ESC)
- Higher-end Electrical Power Steering (EPS)
- Airbag and sensor fusion applications
- Radar applications

A typical aspect of ESC and EPS is the presence of an advanced electrical motor control periphery with special enhancements in the area of pulse width modulations, highly flexible timers, and functional safety. All devices in this family are built around a safety concept based on delayed lock step, targeting an ISO26262 ASIL-D integrity level.

2.3 Features

The following table summarizes the features of the MPC5744P.

Table 2-1. MPC5744P feature summary

Feature	Details
CPU	
Power Architecture	2 x e200z4 in delayed lock step
Architecture	Harvard
Execution speed	0 MHz to 200 MHz (+2% FM)
Embedded FPU	Yes
Core MPU	24 regions
Instruction Set PPC	No
Instruction Set VLE	Yes
Instruction cache	8 KB, EDC
Data cache	4 KB, EDC
Data local memory	64 KB, ECC
System MPU	Yes (16 regions)
Buses	
Core bus	AHB, 32-bit address, 64-bit data, e2e ECC
Internal periphery bus	32-bit address, 32-bit data
Crossbar	
Master x slave ports	4 x 5
Memory—see Table 2-2 for additional details	
Code/data flash memory	**2.5 MB**, ECC, RWW
Data flash memory	Supported with RWW
SRAM	**384 KB**, ECC
Overlay access to SRAM from Flash Memory Controller	Yes
Modules	
Interrupt controller	32 interrupt priority levels, 16 SW programmable interrupts
PIT	1 module with 4 channels
System Timer Module (STM)	1 module with 4 channels
Software Watchdog Timer (SWT)	Yes
eDMA	32 channels, in delayed lock step
FlexRay	1 module with 64 message buffer, dual channel
FlexCAN	3 modules with 64 message buffer

Table continues on the next page...

Table 2-1. MPC5744P feature summary (continued)

Feature	Details
LINFlexD (UART and LIN with DMA support)	2 modules
Clockout	Yes
Fault Control and Collection Unit (FCCU)	Yes
Cross Triggering Unit (CTU)	2 modules
eTimer	3 modules with 6 channels
FlexPWM	2 modules with 4 x (2+1) channels
Analog-to-digital converter (ADC)	4 modules with 12-bit ADC, each with 16 channels (25 external channels including shared channels plus internal channels)
Sine-wave generator (SGEN)	32 point
SPI	4 modules As many as 8 chip selects
CRC Unit	Yes
SENT	2 modules with 2 channels
Interprocessor serial link interface (SIPI)	Yes
Junction temperature sensor	Yes (replicated module)
Digital I/Os	≥ 16
Peripheral register protection	Yes
Ethernet	Yes
Error Injection Module (EIM)	Yes
Supply	
Device Power Supply	3.3 V with external ballast transistor 3.3 V with external 1.25 V low drop-out (LDO) regulator
ADC Analog Reference voltage	3.15 V to 5.5 V
Clocking	
Phase Lock Loop (PLL)	1 x PLL and 1 coupled FMPLL
Internal RC Oscillator	16 MHz
External Crystal Oscillator	8 MHz to 40 MHz
Low power modes	
HALT and STOP	Yes
Debug	
Nexus	Level 3+, MDO and Aurora interface
Package	
LQFP	144 pins, 0.5 mm pitch, 20 mm x 20 mm outline
MAPBGA	257 MAPBGA, 0.8 mm pitch, 14 mm x 14 mm outline
Temperature	
Temperature range (junction)	-40°C to +150°C, option for 165°C
Ambient temperature range (LQFP)	-40°C to +125°C, 135°C option (with 165°C junction option)
Ambient temperature range (BGA)	-40°C to +125°C, 135°C option (with 165°C junction option)

Table 2-2. Flash memory and SRAM sizes of MPC5744P, MPC5743P, MPC5742P, and MPC5741P

Part number	Flash memory	SRAM
MPC5744P	2.5 MB	384 KB
MPC5743P	2.0 MB	256 KB
MPC5742P	1.5 MB	192 KB
MPC5741P	1.0 MB	128 KB

The MPC5744P architecture is based on the Automotive Core Platform (ACP) developed as part of the C55 Template.

This architecture is centered around the scalable e200 family, the AMBA standards, and Freescale’s IP standards.

- AMBA is a 64-bit bus supporting a minimum bus cycle of 1 clock (one for address and one for data phase with limited pipelining). AMBA masters are implemented within the flexible crossbar and a very flexible synchronous protocol converter AMBA-to-IPS gasket (AIPS) is available.
- IPS is a 32-bit single master protocol that needs a minimum of 3 AMBA cycles for unbuffered write and minimum of 2 clocks for periphery read.

Redundancy is implemented for those blocks requiring it to ensure high diagnostic coverage for the key IP (for example: the e200z4 core, the DMA controller, and the AIPS bus bridge with the peripheral subsystems).

The architecture aims to guarantee maximum throughput of 200MHz.

2.3.1 Software debug and calibration

MPC5744P includes many features to facilitate software development. These features include:

- Nexus code execution trace on the main e200z4251n3 core
- Nexus data trace on all bus masters (DMA, Interprocessor bus, CPUs, and others)
- High speed Nexus trace output port consisting of two lanes of Aurora (LVDS) operating up to 1.25 Gbit/s

2.3.2 Compatibility with MPC5643L

The MPC5744P is compatible in important ways with the MPC5643L, a predecessor device in 90 nm CMOS technology. For customers migrating to the MPC5744P, this high degree of compatiblity enables re-use and reduces unnecessary changes. The following table summarizes key aspects of the compatibility.

Table 2-3. MPC5744P Compatibility with MPC5643L

Topic	Remarks for MPC5744P
Package: 144 LQFP	Delivered in the same 144 LQFP with compatible pinout.
Package: 257 MAPBGA	Delivered in the same 257 MAPBGA package. The pinout is compatible, but with limitations due to the new LFAST and Aurora interfaces. Board changes are required.
Power supply concept	Supports an external ballast transistor with internal regulation that allows compatible use. In addition, offers an option with an external supply for the 1.25 V core voltage.
Supply monitoring	The same low voltage monitors for the 1.25 V core voltage and the 3.3 V voltage domains are offered. Both devices offer a high voltage detection for the 1.25 V domain.
Power supply pins	All power supply pins in 144 LQFP package are the same between the two parts, with an exception: the VDD_HV_REG_x pins are replaced with pads for reset/GPIO functionality because the MPC5744P does not offer an internal ballast supply option. When the MPC5744P is soldered into a board designed for the MPC5643L that is unchanged, these GPIOs must not be configured as output.
I/O multiplexing	All I/O multiplexing options in the 144 LQFP package are maintained to allow use of the same peripheral interfaces on the same pins. The MPC5744P adds additional multiplexing for new functionality.
ADC channels and reference voltage	The same pins can be mapped to the same ADC module channels. The ADC are 5 V input capable and have separate reference voltages. The use of ADC pads as inputs on the MPC5744P follows a 5 V I/O specification. In contrast, the specification is 3.3 V on the MPC5643L.
External hardware for safety	The same external components are required as on the MPC5643L: • monitoring of the Error Out interface • external watchdog functionality (periodic communication) • external 3.3 V HVD

2.3.3 Block Diagram

The following figure is a top-level diagram that shows the functional organization of the system.

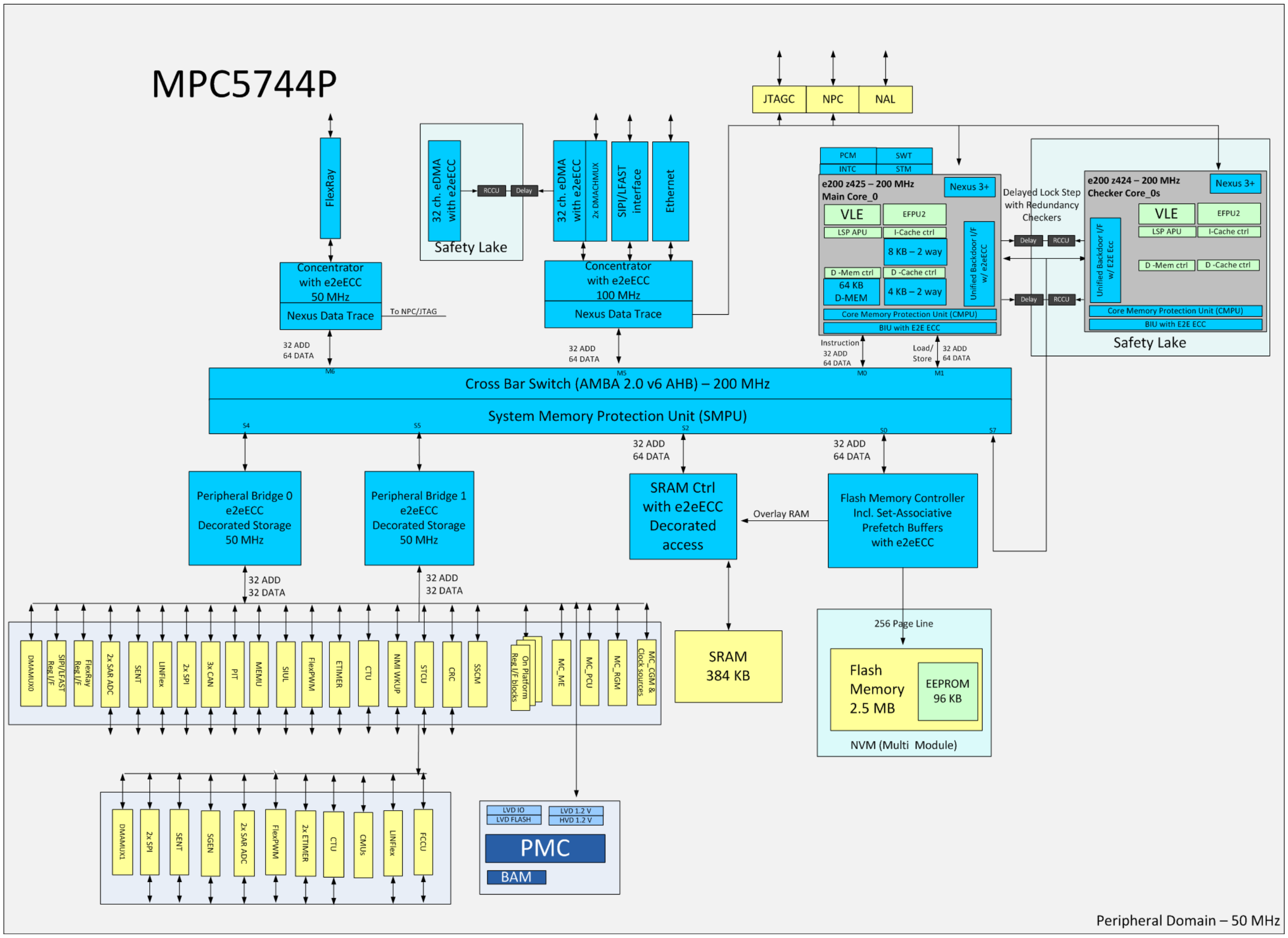


Figure 2-1. System Block Diagram

Chapter 3
Embedded Memories

3.1 Overview

The embedded memory architecture for MPC5744P includes the features:

- Onboard SRAM, including system SRAM, local data memory for each processor core, and overlay SRAM
- Onboard system ECC flash memory
- End-to-end ECC error detection and correction
- Built-in flash memory security features including censorship

3.2 SRAM

3.2.1 System SRAM

MPC5744P includes 384 KB general-purpose on-chip ECC SRAM. The SRAM can be configured for either 0- or 1-wait state read operation latency using the PRCR1 register in the SRAM controller.

- See the SRAM controller memory map and register definition for details on configuration of the SRAM controller.
- See the Memory Map chapter for the MPC5744P SRAM map.

A portion of the system SRAM can be used as the overlay SRAM. *The overlay SRAM* feature included in MPC5744P is part of a comprehensive set of calibration and debug features. Overlay SRAM can be mapped over specific regions of on-chip flash memory so that any access to an overlaid flash address is routed to the overlay SRAM instead. This enables calibration of constant data without requiring additional external RAMs and calibration memory interfaces.

NOTE

Overlay SRAM is normally used for system development purposes only.

Timing for calibration accesses to a particular overlay SRAM are the same as access to the underlying flash if the overlay SRAM is not being used as a destination for trace streaming at the same time.

The overlay function is implemented using registers in the flash memory controller.

1. Using PFLASH Calibration Region Descriptors (PFCRDn) define one or more (up to 32) overlay regions.
2. Enable individual regions using the appropriate bits in the PFLASH Remap Descriptor Enable Register (PFCRDE).
3. Enable overlay using the PFCRCR[GRMEN] field.

See PFLASH calibration remap support for details on memory overlay (remapping).

3.2.2 Processor core local SRAM

MPC5744P includes local data SRAM (D-MEM) for the processor core to provide enhanced performance.

- Each processor core has 64 KB of data SRAM.

Each area of core SRAM is accessible by other processor cores and bus mastering peripheral devices. See the Memory Map for details.

3.3 Flash memory

Flash memory on MPC5744P consists of a flash memory controller and a flash memory array module. The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the device crossbar switch. The following figure shows the memory architecture.

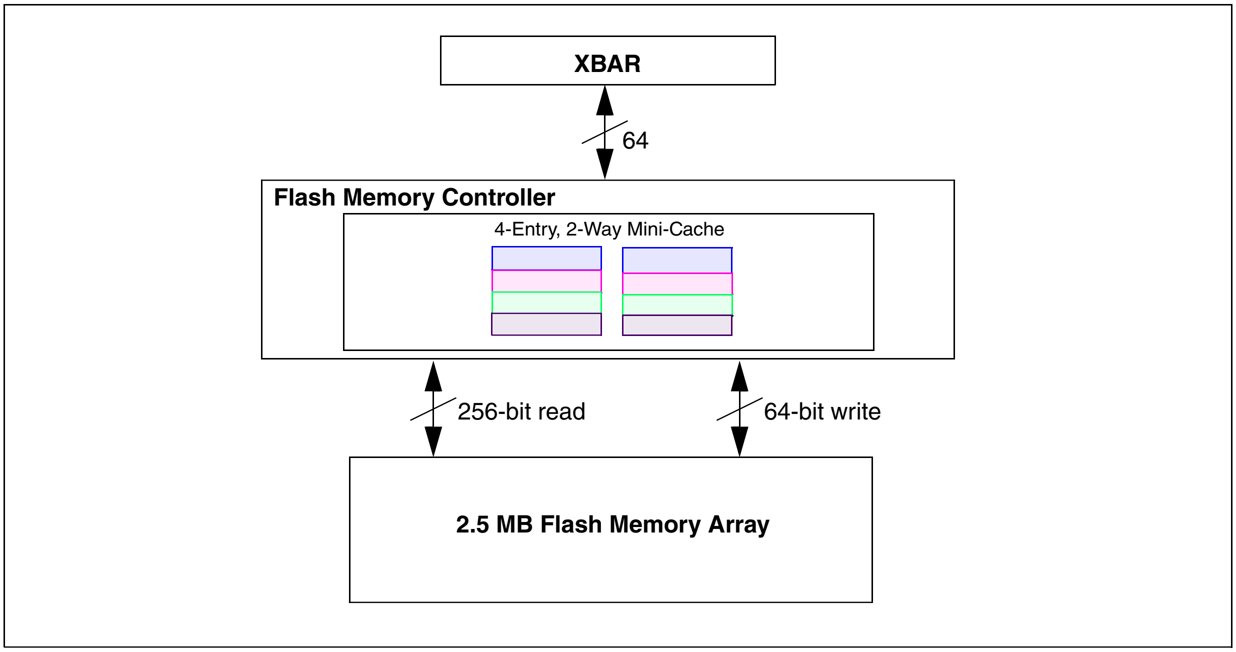


Figure 3-1. Device flash memory block diagram

3.3.1 Flash memory controller

The MPC5744P flash controller acts as an interface between the system bus and the flash array and serves as the interface to the on-chip overlay RAM.

The flash controller contains a 4-entry, 2-way set-associative mini-cache that delivers flash read data with a zero-wait state response on lines that reside in the cache. Each entry contains one flash page, a 256-bit (32-byte) memory value. Read requests that miss the cache generate the needed flash array access.

The flash memory controller contains configuration registers which manage flash functionality such as read buffering in the mini-cache, access control, calibration RAM overlay remapping, and read wait state management of the flash.

See the Flash Memory Controller chapter for details.

3.3.2 Flash memory array

The MPC5744P on-chip flash provides programmable, non-volatile flash memory. The non-volatile memory (NVM) can be used for instruction storage, data storage, or both. The flash memory module interfaces the system bus to a dedicated flash memory array controller. It supports a 64-bit data bus width at the system bus port, and a 256-bit read data interface to flash memory. The module contains a 2-way set-associative mini-cache.

3.3.2.1 Features

- 2.5MB of Flash in unique multi partitioned hard macro
- Flash partitioning
  - 4x 16 KB in partition 0/1 (2x blocks EEPROM emulation enabled)
  - 2x 32 KB in partition 2/3 (EEPROM emulation enabled)
  - 6x 64 KB in partition 4/5
  - 8x 256 KB in partition 6/7
- Support for reading-while-writing when the accesses are to different partitions.
- Flash protection
  - Write protection and OTP available for dedicated blocks.
- Test information stored in a non-volatile UTest block which will be OTP.
- Erase suspend, program suspend and erase-suspended program all supported.
- 256 KB address space

3.3.2.2 UTest memory space

MPC5744P contains a 16 KB area of One-Time Programmable (OTP) flash memory for storage of test information and device configuration data. See Chapter 5, Memory Map for the MPC5744P UTEST flash memory map.

3.4 End-to-end Error Correction Code (e2eECC)

To support market requirements related to improved functional and transient fault detection capabilities, this family of automotive microcontrollers includes end-to-end ECC support. This end-to-end ECC (aka e2eECC) is structurally different than traditional "ECC at memory" functionality since it provides for robust error detection capabilities from one endpoint of an information transfer to another endpoint with temporary information storage in an intermediate component(s). While memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect and/or correct errors which have occurred on stored data in the memory, e2eECC instead performs generation of error protection codes at the source of data generation, sending the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master, and performs a check of data integrity using the previously stored error protection codes at a data memory when a read of stored information is requested by a bus master. The intermediate storage may transform the generated error protection codes into another format for storage and then may regenerate the codes for provision when a request is made to read the stored information, or it may simply store the original protection codes unaltered, depending on the particular unit.

Additionally, the error protection codes are generated based on more than just the data associated with a storage location in order to protect additional information associated with an access. In particular, address information corresponding to the access location of the stored information is combined with the store data at the data source to generate error protection codes which cover certain types of addressing errors which may occur in the system interconnect or in the memory unit. Checking of the error protection codes may be done at the memory unit on a store to ensure that no corruption of address or data information has occurred while the request has transitioned through the device from the bus master source, as well as to ensure that within the storage memory, address decoding was performed properly (although not all address decoding errors can be detected this way), or it may simply store the received data and protection codes at the address it receives. On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location(s), and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory, or in the retrieved data, thus ensuring that the address sent for the request was not corrupted, that the addressed location was actually accessed (to the extent it is possible to ensure), and that the stored data was error free, to the extend the ECC coding scheme is able to detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust that the previous implementation of locally generated and checked/corrected error protection at each memory unit.

A comparison of the traditional and e2eECC approaches is shown in the following table.

Table 3-1. ECC comparison of data write then read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data + checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave

Table continues on the next page...

Table 3-1. ECC comparison of data write then read sequence (continued)

Traditional ECC	End-to-End ECC (e2eECC)
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

The scope of differences in the operations "covered" by the ECC checks is readily apparent. Thus, the e2eECC concept provides improved fault detection capabilities in two important aspects:

1. The entire data transaction, from the initiating bus master, through the entire platform crossbar steering mechanism to the destination slave target is covered during write accesses. Likewise, a read is checked from the initiating bus master, through the crossbar steering mechanism, through the actual memory read and transmission of the data plus checkbits back to the bus master, where the integrity and correctness of the entire transaction is checked.
2. The selected ECC provides protection of both the address field as well as the data field, again for improved fault coverage.

Additionally, this particular structure also provides a significant implementation improvement. The traditional ECC at the memory approach places the checkbit decode and error syndrome generation with the error/no_error state affecting whether the system bus transfer must be stalled (to correct a single bit error or report a noncorrectable event) or is allowed to complete (for error free transfers) in a critical timing arc which often sets the upper limit of operating frequency of the microcontroller. With the e2eECC scheme, the checkbit decode and error syndrome logic is located in the requesting bus master, where there are generally more degrees of implementation freedom and the error/no_error state determination is removed from the system bus cycle termination logic, producing an improved timing arc and generally, higher operating speeds.

Errors which occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in corrupted read data or checkbits being returned to a requestor. While not all possible errors can be detected this way, this approach provided a substantial improvement versus traditional methods. Additional on-line diagnostics and/or additional hardware should be able to catch a majority of the errors not covered directly by the e2eECC scheme.

Chapter 4
Signal Description

4.1 Production packages

The device has the following production packages:

- 144LQFP
- 257MAPBGA

The following table shows features/pins provided by the 257MAPBGA compared to the 144LQFP.

Table 4-1. MPC5744P features differing by package

Feature	144LQFP	257MAPBGA
FlexPWM1	A[0-2]/B[0-2]	A[0-3]/B[0-3]/X[0-3]/Fault[0:3]
eTimer2	ETC2-5	ETC0-5
Supplies	3.3V (IO): 4xV_{DD}, 4xV_{SS} 3.3V (PMU): 1xV_{DD} 3.3V (OSC): 1xV_{DD}[1] 3.3V (flash): 1xV_{DD}[1] 1.25V (core): 6xV_{DD}, 8xV_{SS} 1.25V (PLL): 1xV_{DD},[1] 1xV_{SS} 1.25V (flash): 1xV_{DD}[1, 2] 3.3V/5V (ADR): 2xV_{DD}, 2xV_{SS} 3.3V (ADV): 1xV_{DD}, 1xV_{SS}	Additional supplies beyond 144LQFP: 3.3V (IO): 5xV_{DD}, 15xV_{SS} 1.25V (LFAST): 1xV_{DD} 1.25V (core): 18xV_{DD}, 17xV_{SS}
GPIO	79 GPIO 26 GPI	Additional GPIO beyond 144LQFP: 33 GPIO 3 GPI 2 LFAST differential pads Aurora Interface: 2x transmit, 1x receive (reference clock)
DSPI3	No external signals	CS0-3

Table continues on the next page...

Table 4-1. MPC5744P features differing by package (continued)

Feature	144LQFP	257MAPBGA
CTU external trigger(s)	CTU0	CTU0 CTU1
ADCs[3]	• 22 analog pads assigned to ADC0, ADC1, ADC2, and ADC3 • Shared channels between ADC0/ADC1, ADC0/ADC2, and ADC1/ADC3	• 25 analog pads assigned to ADC0, ADC1, ADC2, and ADC3 • Shared channels between ADC0/ADC1, ADC0/ADC2, ADC1/ADC3, and ADC2/ADC3
SIPI/ LFAST interface	No	Yes
Ethernet	No	Yes
Nexus	Yes (MDO interface)	Yes (MDO interface)
Nexus Aurora port	No	Yes (TX0/TX0_P/TX1/TX1_P/CLK/CLK_P)

1. V_{SS} connected to Core Ground on 144LQFP
2. 1.25V flash memory supply connected to Core Supply on 144LQFP (93)
3. For all packages, internal analog sources/supplies can be connected to ADC0, ADC1, ADC2, and ADC3.

Case numbers for outline drawings of each package are provided in the MPC5744P Data Sheet.

4.2 Package pinouts and ballmap

The following figures show the LQFP pinout and the BGA ballmap.

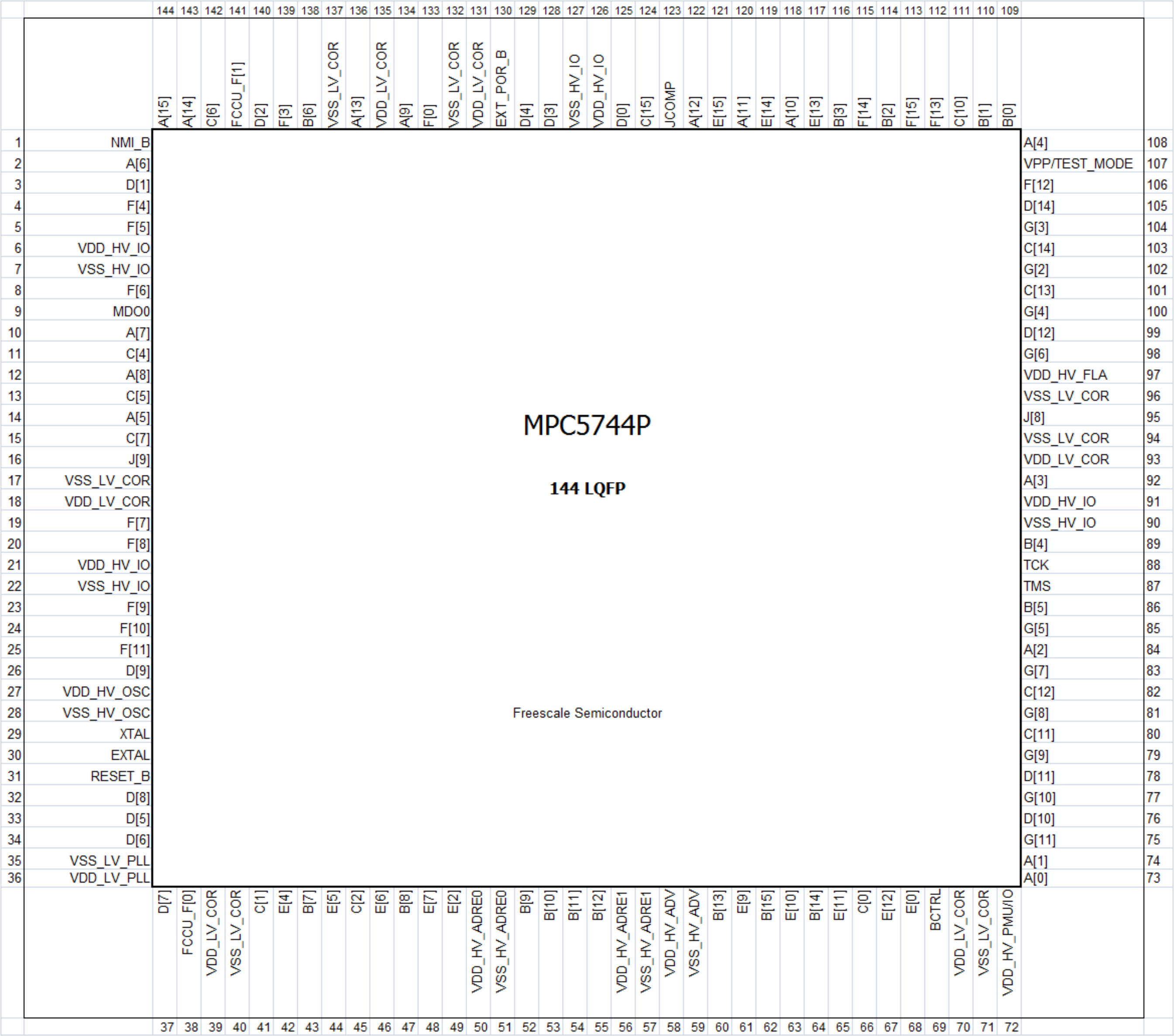
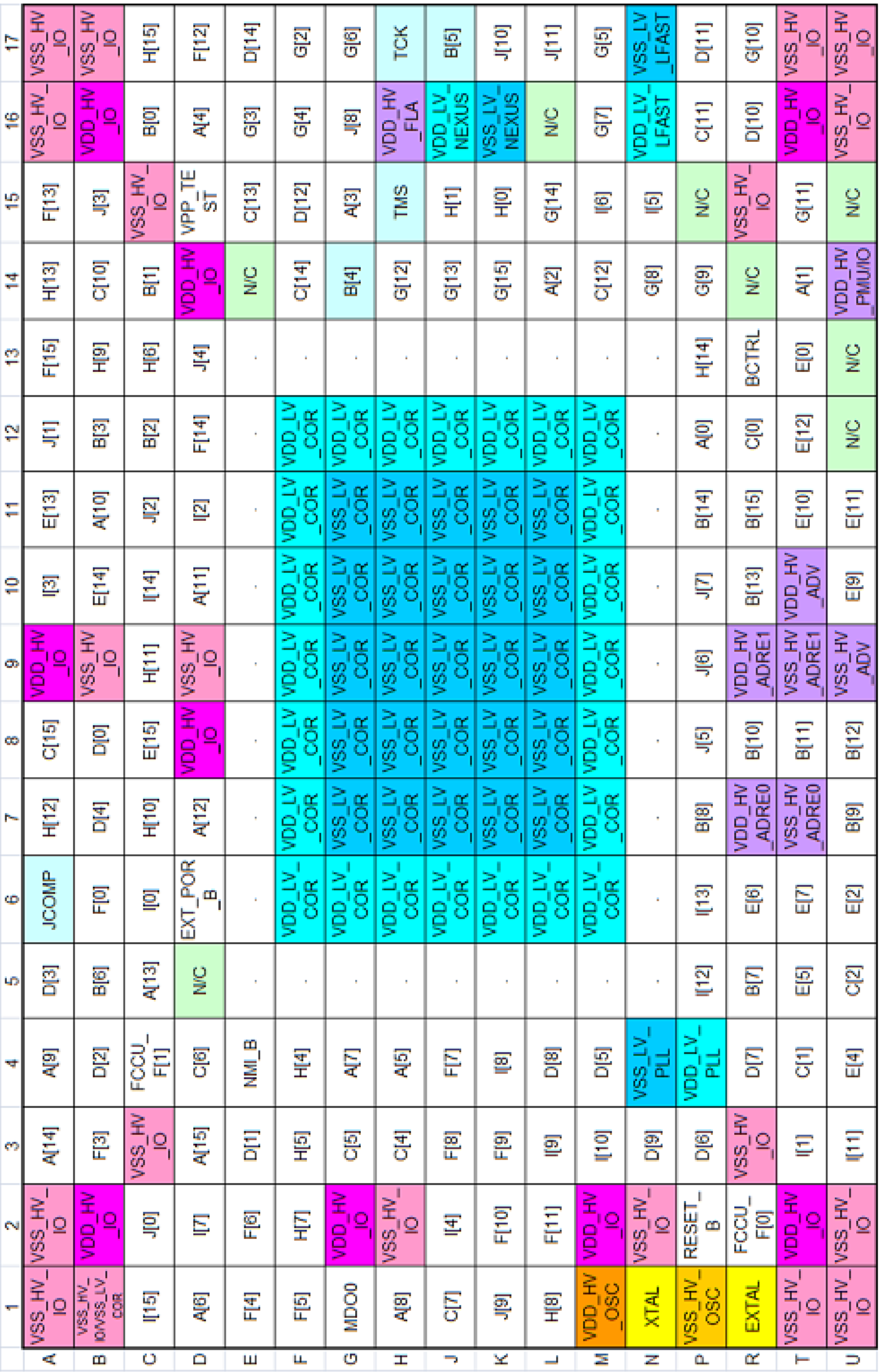


Figure 4-1. 144LQFP pinout



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	VSS_HV_IO	VSS_HV_IO	A[14]	A[9]	D[3]	JCOMP	H[12]	C[15]	VDD_HV_IO	I[3]	E[13]	J[1]	F[15]	H[13]	F[13]	VSS_HV_IO	VSS_HV_IO
B	VSS_HV_IO/VSS_LV_COR	VDD_HV_IO	F[3]	D[2]	B[6]	F[0]	D[4]	D[0]	VSS_HV_IO	E[14]	A[10]	B[3]	H[9]	C[10]	J[3]	VDD_HV_IO	VSS_HV_IO
C	I[15]	J[0]	VSS_HV_IO	FCCU_F[1]	A[13]	I[0]	H[10]	E[15]	H[11]	I[14]	J[2]	B[2]	H[6]	B[1]	VSS_HV_IO	B[0]	H[15]
D	A[6]	I[7]	A[15]	C[6]	N/C	EXT_POR_B	A[12]	VDD_HV_IO	VSS_HV_IO	A[11]	I[2]	F[14]	J[4]	VDD_HV_IO	VPP_TEST	A[4]	F[12]
E	F[4]	F[6]	D[1]	NMI_B	.	.	.	.	.	.	.	.	.	N/C	C[13]	G[3]	D[14]
F	F[5]	H[7]	H[5]	H[4]	.	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	.	C[14]	D[12]	G[4]	G[2]
G	MDO0	VDD_HV_IO	C[5]	A[7]	.	VDD_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VDD_LV_COR	.	B[4]	A[3]	J[8]	G[6]
H	A[8]	VSS_HV_IO	C[4]	A[5]	.	VDD_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VDD_LV_COR	.	G[12]	TMS	VDD_HV_FLA	TCK
J	C[7]	I[4]	F[8]	F[7]	.	VDD_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VDD_LV_COR	.	G[13]	H[1]	VDD_LV_NEXUS	B[5]
K	J[9]	F[10]	F[9]	I[8]	.	VDD_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VDD_LV_COR	.	G[15]	H[0]	VSS_LV_NEXUS	J[10]
L	H[8]	F[11]	I[9]	D[8]	.	VDD_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VSS_LV_COR	VDD_LV_COR	.	A[2]	G[14]	N/C	J[11]
M	VDD_HV_OSC	VDD_HV_IO	I[10]	D[5]	.	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	VDD_LV_COR	.	C[12]	I[6]	G[7]	G[5]
N	XTAL	VSS_HV_IO	D[9]	VSS_LV_PLL	.	.	.	.	.	.	.	.	.	G[8]	I[5]	VDD_LV_LFAST	VSS_LV_LFAST
P	VSS_HV_OSC	RESET_B	D[6]	VDD_LV_PLL	I[12]	I[13]	B[8]	J[5]	J[6]	J[7]	B[14]	A[0]	H[14]	G[9]	N/C	C[11]	D[11]
R	EXTAL	FCCU_F[0]	VSS_HV_IO	D[7]	B[7]	E[6]	VDD_HV_ADRE0	B[10]	VDD_HV_ADRE1	B[13]	B[15]	C[0]	BCTRL	N/C	VSS_HV_IO	D[10]	G[10]
T	VSS_HV_IO	VDD_HV_IO	I[1]	C[1]	E[5]	E[7]	VSS_HV_ADRE0	B[11]	VSS_HV_ADRE1	VDD_HV_ADV	E[10]	E[12]	E[0]	A[1]	G[11]	VDD_HV_IO	VSS_HV_IO
U	VSS_HV_IO	VSS_HV_IO	I[11]	E[4]	C[2]	E[2]	B[9]	B[12]	VSS_HV_ADV	E[9]	E[11]	N/C	N/C	VDD_HV_PMU/IO	N/C	VSS_HV_IO	VSS_HV_IO

Figure 4-2. 257MAPBGA ballmap

4.3 Pin/ball descriptions

The following sections provide signal descriptions and related information about the functionality and configuration of the device. Note that this section is under development.

4.3.1 Pin/ball startup and reset states

The following table provides startup state and reset state information for device pins/balls.

The startup state and subsequent states of the following pins/balls cannot be configured by the user:

- JCOMP
- TMS
- TCK
- XTAL/EXTAL
- FCCU_F[0] and FCCU_F[1]
- EXT_POR_B
- RESET_B

The user can configure the state after reset of the following pins/balls by programming the applicable MSCRs/IMCRs:

- GPIOs
- Analog inputs
- TDI
- TDO
- NMI_B
- FAB
- ABS[0]
- ABS[2]

Table 4-2. Pin/ball startup and reset states

Pin/ball	Startup state[1]	State during reset	State after reset	144LQFP	257MAPBGA
GPIOs	hi-z	hi-z	hi-z	Note	Note[2]
Analog inputs[3]	hi-z	hi-z	hi-z	Note[2]	Note[2]
JCOMP (TRST)	hi-z	input, weak pull-down	input, weak pull-down	Note	Note[4]
TDI	hi-z	input, weak pull-up	input, weak pull-up	Note[4]	Note[4]
TDO	hi-z	output, hi-z	output, hi-z	Note[4]	Note[4]
TMS	hi-z	input, weak pull-up	input, weak pull-up	Note[4]	Note[4]

Table continues on the next page...

Table 4-2. Pin/ball startup and reset states (continued)

Pin/ball	Startup state[1]	State during reset	State after reset	144LQFP	257MAPBGA
TCK[5]	hi-z	input, weak pull-up	input, weak pull-up	Note[4]	Note[4]
XTAL/EXTAL	hi-z	hi-z	hi-z	Note[4]	Note[4]
FCCU_F[0][5]	hi-z	input, hi-z	output/input, hi-z	38	R2
FCCU_F[1][5]	hi-z	input, hi-z	output/input, hi-z	141	C4
EXT_POR_B	hi-z	input, weak pull-down	input, weak pull-down	Note[4]	Note[4]
RESET_B	hi-z	input, weak pull-down	input, weak pull-down	Note[4]	Note[4]
NMI_B	hi-z	input, weak pull-up	input,weak pull-up	Note[4]	Note[4]
FAB	hi-z	input, weak pull-down	input, weak pull-down	Note[4]	Note[4]
ABS[2]	hi-z	input, weak pull-down	input, weak pull-down	Note[4]	Note[4]
ABS[0]	hi-z	input, weak pull-down	input, weak pull-down	Note[4]	Note[4]

1. Startup state is exited when the core and high-voltage supplies reach minimum levels as defined in the Power Management chapter.
2. See Generic pins/balls.
3. Not all non-supply or reference pins on the device are explicitly defined in this table.
4. See System pins/balls.
5. This pin/ball is dedicated to and directly connected to a peripheral module pin.

4.3.2 Power supply and reference voltage pins/balls

Table 4-3. Power supply and reference voltage pins/balls

Supply			Package	
Symbol	Type	Description	144LQFP	257MAPBGA
$V_{DD\_LV\_COR}$	Power	Low voltage power Supply	18 39 70 93 131 135	F6 F7 F8 F9 F10 F11 F12 G6 G12 H6 H12 J6 J12 K6 K12 L6

Table continues on the next page...

Table 4-3. Power supply and reference voltage pins/balls (continued)

Supply			Package	
Symbol	Type	Description	144LQFP	257MAPBGA
				L12 M6 M7 M8 M9 M10 M11 M12
$V_{SS\_LV\_COR}$	Ground	Low voltage ground. PLL Ground is also connected to low voltage ground for core logic on 144LQFP (pin 35).	17 35 40 71 94 96 132 137	B1 G7 G8 G9 G10 G11 H7 H8 H9 H10 H11 J7 J8 J9 J10 J11 K7 K8 K9 K10 K11 L7 L8 L9 L10 L11
$V_{DD\_LV\_PLL}$	Power	PLL low voltage Supply	36	P4
$V_{SS\_LV\_PLL}$	Ground	PLL low voltage Ground	35	N4
$V_{DD\_HV\_IO}$	Power	High voltage Power Supply for I/O	6	A9

Table continues on the next page...

Table 4-3. Power supply and reference voltage pins/balls (continued)

Supply			Package	
Symbol	Type	Description	144LQFP	257MAPBGA
			21 72 91 126	B2 B16 D8 D14 G2 M2 T2 T16 U14
$V_{SS\_HV\_IO}$	Ground	High voltage Ground Supply for I/O	7 22 90 127	A1 A2 A16 A17 B1 B9 B17 C3 C15 D9 H2 N2 R3 R15 T1 T17 U1 U2 U16 U17
$V_{DD\_HV\_PMU}$ $V_{DD\_HV\_PMU\_AUX}$	Power	PMU high voltage Supply	72	U14
$V_{DD\_HV\_OSC}$	Power	Power Supply for the oscillator	27	M1
$V_{SS\_HV\_OSC}$	Ground	Ground Supply for the oscillator	28	P1
$V_{DD\_HV\_FLA}$	Power	Power Supply and decoupling pin for flash memory	97	H16
$V_{DD\_HV\_ADV}$	Power	High voltage Supply for ADC, TSENS, SGEN (3.3 V)	58	T10
$V_{SS\_HV\_ADV}$	Ground	High voltage Ground for ADC	59	U9
$V_{DD\_HV\_ADRE0}$	Supply	High voltage Supply for digital portion of ADC pads	50	R7

Table continues on the next page...

Table 4-3. Power supply and reference voltage pins/balls (continued)

Supply			Package	
Symbol	**Type**	**Description**	**144LQFP**	**257MAPBGA**
		Voltage reference of ADC/TSENS High voltage Supply for ADC0 pads and shared pads for ADC0/1. See ADC Configuration for more information.		
$V_{SS\_HV\_ADRE0}$	Ground	High voltage Ground for digital portion of ADC pads Voltage reference Ground of ADC/TSENS High voltage Ground for ADC0 pads and shared pads for ADC0/1. See ADC Configuration for more information.	51	T7
$V_{DD\_HV\_ADRE1}$	Supply	High voltage Supply for digital portion of ADC pads Voltage reference of ADC/TSENS High voltage Supply for ADC1 pads, shared pads for ADC1/3, and shared pads for ADC2/3. See ADC Configuration for more information.	56	R9
$V_{SS\_HV\_ADRE1}$	Ground	High voltage Ground for digital portion of ADC pads Voltage reference Ground of ADC/TSENS High voltage Ground for ADC1 pads, shared pads for ADC1/3, and shared pads for ADC2/3. See ADC Configuration for more information.	57	T9
$V_{DD\_LV\_LFAST}$	Supply	LFAST PLL low voltage Supply	—	N16
$V_{SS\_LV\_LFAST}$	Ground	LFAST PLL low voltage Ground	—	N17
$V_{DD\_LV\_NEXUS}$	Supply	Aurora LVDS Supply	—	J16
$V_{SS\_LV\_NEXUS}$	Ground	Aurora LVDS Ground	—	K16

4.3.3 System pins/balls

The following table contains information about system pin functions for the devices.

Table 4-4. System pins/balls

Symbol	Type	Description	144LQFP	257MAPBGA
NMI_B	Input	Non-maskable Interrupt	1	E4
XTAL	Input	Crystal Oscillator/External Clock Input	29	N1
EXTAL	Input	Input of the oscillator amplifier circuit	30	R1
RESET_B	Input	Functional Reset	31	P2
EXT_POR_B	Input	External Power On Reset	130	D6
VPP_TEST[1]	Input	SoC Test Mode	107	D15
JCOMP	Input	JTAGC, JTAG Compliance Enable	123	A6
TCK	Input	JTAGC, Test Clock Input	88	H17
TMS	Input	JTAGC, Test Mode Select	87	H15
TDO	Output	JTAGC, Test Data Out	89	G14

Table continues on the next page...

Table 4-4. System pins/balls (continued)

Symbol	Type	Description	144LQFP	257MAPBGA
TDI	Input	JTAGC, Test Data Input	86	J17
MDO[0]	Output	NEXUS, Message data out pins; reflects the state of the internal power on reset signal until RESET is negated	9	G1
MDO[3:1]	Output	NEXUS, Message data out pins	4,5,8	E1, F1, E2
EVTO	Output	NEXUS, Event Out Pin	24	K2
EVTI	Input	NEXUS, Event In Pin	25	L2
MCKO	Output	NEXUS, Message clock out pin	19	J4
MSEO[1:0]	Output	NEXUS, Message Start/End out pin	20, 23	J3, K3
RDY_B	Output	NEXUS, Read/Write Transfer completed	—	J2
			16	K1
BCTRL	Output	Base control signal of external npn ballast	69	R13
J[11], J[10]	--	FSL Factory Test[2]	—	L17, K17

1. VPP_TEST must be connected to ground.
2. Do not connect on the board.

4.3.4 LVDS pins/balls

The following tables contain information on LVDS pin functions for the devices.

Table 4-5. SIPI LFAST LVDS pin descriptions

Functional block	Port pin	Signal	Signal description	Direction	257MAPBGA
SIPI LFAST[1, 2]	I[5]	SIPI_TXN	Interprocessor Bus LFAST, LVDS Transmit Negative Terminal	O	N15
	C[12]	SIPI_TXP	Interprocessor Bus LFAST, LVDS Transmit Positive Terminal	O	M14
	I[6]	SIPI_RXN	Interprocessor Bus LFAST, LVDS Receive Negative Terminal	I	M15
	G[7][3]	SIPI_RXP	Interprocessor Bus LFAST, LVDS Receive Positive Terminal	I	M16

1. DRCLK and TCK/DRCLK usage for SIPI LFAST are described in the reference manual's SIPI LFAST chapters.
2. For the MSCR SSS value of the port pin, see Table 1.
3. The 144LQFP package has G[7] and C[12] but no SIPI LFAST functionality.

CAUTION

SIPI LFAST pins are muxed with GPIOs. Do not use GPIO and SIPI LFAST functionality in parallel.

Table 4-6. Aurora LVDS pin descriptions

Functional block	Pad	Signal	Signal description	Direction	257MAPBGA[1]
Nexus Aurora High Speed Trace	G[12]	TX0P	Nexus Aurora High Speed Trace Lane 0, LVDS Positive Terminal	O	H14
	G[13]	TX0N	Nexus Aurora High Speed Trace Lane 0, LVDS Negative Terminal	O	J14
	G[14]	TX1P	Nexus Aurora High Speed Trace Lane 1, LVDS Positive Terminal	O	L15
	G[15]	TX1N	Nexus Aurora High Speed Trace Lane 1, LVDS Negative Terminal	O	K14
	H[0]	CLKP	Nexus Aurora High Speed Trace Clock, LVDS Positive Terminal	I	K15
	H[1]	CLKN	Nexus Aurora High Speed Trace Clock, LVDS Negative Terminal	I	J15

1. Nexus Aurora High Speed Trace is available only on the 257MAPBGA.

4.3.5 Generic pins/balls

The I/O signal descriptions for the device are in the following table. It contains the port definition, multiplexing, direction, pad type, and package pin/ball numbers for each I/O pin on the device.

MSCR registers are used for alternative (ALT) mode selection and programming of pad control options.

IMCR registers are used to configure input muxing by peripheral. See Peripheral input muxing for details.

See Table 4-16 for the MSCR register address map and Table 4-17 for the IMCR register address map.

Table 4-7. Pin muxing

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
A[0]	MSCR[0]	0000 (Default)[2]	GPIO[0]	SIUL2-GPIO[0]	General Purpose IO A[0]	I/O	73	P12
		0001	ETC0	eTimer_0	eTimer_0 Input/Output Data Channel 0	I/O		
		0010	SCK	DSPI2	DSPI 2 Serial Clock (output)	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[48]	0001	SCK	DSPI2	DSPI 2 Serial Clock (input)	I/O		
	IMCR[59]	0010	ETC0	eTimer_0	eTimer_0 Input Data Channel 0	I/O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[173]	0001	REQ0	SIUL2	SIUL2 External Interrupt 0	I		
A[1]	MSCR[1]	0000 (Default)	GPIO[1]	SIUL2-GPIO[1]	General Purpose IO A[1]	I/O	74	T14
		0001	ETC1	eTimer_0	eTimer_0 Input/Output Data Channel 1	I/O		
		0010	SOUT	DSPI2	DSPI 2 Serial Data Out	O		
		0011-1111	—	Reserved	—	—		
	IMCR[60]	0010	ETC1	eTimer_0	eTimer_0 Input Data Channel 1	I/O		
	IMCR[174]	0001	REQ1	SIUL2	SIUL2 External Interrupt Source 1	I		
A[2]	MSCR[2]	0000 (Default)	GPIO[2]	SIUL2-GPIO[2]	General Purpose IO A[2]	I/O	84	L14
		0001	ETC2	eTimer_0	eTimer_0 Input/Output Data Channel 2	I/O		
		0010	—	Reserved	—	—		
		0011	A3	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 3	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[169]	0000 (Default)	ABS0	MC_RGM	RGM external boot mode 1	I		
	IMCR[47]	0010	SIN	DSPI2	DSPI 2 Serial Data Input	I		
	IMCR[61]	0010	ETC2	eTimer_0	eTimer_0 Input Data Channel 2	I/O		
	IMCR[97]	0001	A3	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 3	I/O		
	IMCR[175]	0001	REQ2	SIUL2	SIUL2 External Interrupt Source 2	I		
A[3]	MSCR[3]	0000 (Default)	GPIO[3]	SIUL2-GPIO[3]	General Purpose IO A[3]	I/O	92	G15
		0001	ETC3	eTimer_0	eTimer_0 Input/Output Data Channel 3	I/O		
		0010	CS0	DSPI2	DSPI 2 Peripheral Chip Select 0	I/O		
		0011	B3	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 3	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[171]	0000 (Default)	ABS2	MC_RGM	RGM external boot mode 2	I		
	IMCR[62]	0010	ETC3	eTimer_0	eTimer_0 Input Data Channel 3	I/O		
	IMCR[49]	0001	CS0	DSPI2	DSPI 2 Peripheral Chip Select 0	I/O		
	IMCR[98]	0001	B3	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 3	I/O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[176]	0001	REQ3	SIUL2	SIUL2 External Interrupt Source 3	I		
A[4]	MSCR[4]	0000 (Default)	GPIO[4]	SIUL2-GPIO[4]	General Purpose IO A[4]	I/O	108	D16
		0001	ETC0	eTimer_1	eTimer_1 Input/Output Data Channel 0	I/O		
		0010	CS1	DSPI2	DSPI 2 Peripheral Chip Select 1	O		
		0011	ETC4	eTimer_0	eTimer_0 Input/Output Data Channel 4	I/O		
		0100	A2	FlexPWM_1	FlexPWM_1 Channel A Input/ Output 2	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[112]	0001	A2	FlexPWM_1	FlexPWM_1 Channel A Input 2	I/O		
	IMCR[177]	0001	REQ4	SIUL2	SIUL2 External Interrupt Source 4	I		
	IMCR[172]	0000 (Default)	FAB	MC_RGM	RGM Force Alternate Boot Mode	I		
	IMCR[65]	0001	ETC0	eTimer_1	eTimer_1 Input Data Channel 0	I/O		
	IMCR[63]	0011	ETC4	eTimer_0	eTimer_0 Input Data Channel 4	I/O		
A[5]	MSCR[5]	0000 (Default)	GPIO[5]	SIUL2-GPIO[5]	General Purpose IO A[5]	I/O	14	H4
		0001	CS0	DSPI1	DSPI 1 Peripheral Chip Select 0	I/O		
		0010	ETC5	eTimer_1	eTimer_1 Input/Output Data Channel 5	I/O		
		0011	CS7	DSPI0	DSPI 0 Peripheral Chip Select 7	O		
		0100-1111	—	Reserved	—	—		
	IMCR[70]	0001	ETC5	eTimer_1	eTimer_1 Input Data Channel 5	I/O		
	IMCR[178]	0001	REQ5	SIUL2	SIUL2 External Interrupt Source 5	I		
A[6]	MSCR[6]	0000 (Default)	GPIO[6]	SIUL2-GPIO[6]	General Purpose IO A[6]	I/O	2	D1
		0001	SCK	DSPI1	DSPI 1 Serial Clock (output)	I/O		
		0010	ETC2	eTimer_2	eTimer_2 Input/Output Data Channel 2	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[73]	0001	ETC2	eTimer_2	eTimer_2 Input Data Channel 2	I/O		
	IMCR[179]	0001	REQ6	SIUL2	SIUL2 External Interrupt Source 6	I		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
A[7]	MSCR[7]	0000 (Default)	GPIO[7]	SIUL2-GPIO[7]	General Purpose IO A[7]	I/O	10	G4
		0001	SOUT	DSPI1	DSPI 1 Serial Data Out	O		
		0010	ETC3	eTimer_2	eTimer_2 Input/Output Data Channel 3	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[74]	0001	ETC3	eTimer_2	eTimer_2 Input Data Channel 3	I/O		
	IMCR[180]	0001	REQ7	SIUL2	SIUL2 External Interrupt Source 7	I		
A[8]	MSCR[8]	0000 (Default)	GPIO[8]	SIUL2-GPIO[8]	General Purpose IO A[8]	I/O	12	H1
		0001	—	Reserved	—	—		
		0010	ETC4	eTimer_2	eTimer_2 Input/Output Data Channel 4	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[44]	0001	SIN	DSPI1	DSPI 1 Serial Data Input	I		
	IMCR[75]	0001	ETC4	eTimer_2	eTimer_2 Input Data Channel 4	I/O		
	IMCR[181]	0001	REQ8	SIUL2	SIUL2 External Interrupt Source 8	I		
A[9]	MSCR[9]	0000 (Default)	GPIO[9]	SIUL2-GPIO[9]	General Purpose IO A[9]	I/O	134	A4
		0001	CS1	DSPI2	DSPI 2 Peripheral Chip Select 1	O		
		0010	ETC5	eTimer_2	eTimer_2 Input/Output Data Channel 5	I/O		
		0011	B3	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 3	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[76]	0001	ETC5	eTimer_2	eTimer_2 Input Data Channel 5	I/O		
	IMCR[98]	0010	B3	FlexPWM_0	FlexPWM_0 Channel B Input 3	I/O		
	IMCR[83]	0001	FAULT0	FlexPWM_0	FlexPWM_0 Fault Input 0	I		
	IMCR[206]	0011	SENT_RX[1]	SENT_0	SENT 0 Receiver channel 1	I		
A[10]	MSCR[10]	0000 (Default)	GPIO[10]	SIUL2-GPIO[10]	General Purpose IO A[10]	I/O	118	B11
		0001	CS0	DSPI2	DSPI 2 Peripheral Chip Select 0	I/O		
		0010	B0	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 0	I/O		
		0011	X2	FlexPWM_0	FlexPWM_0 Auxiliary Input/ Output 2	I/O		
		0100-1111	—	Reserved	—	—		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[49]	0010	CS0	DSPI2	DSPI 2 Peripheral Chip Select 0	I/O		
	IMCR[89]	0001	B0	FlexPWM_0	FlexPWM_0 Channel B Input 0	I/O		
	IMCR[96]	0001	X2	FlexPWM_0	FlexPWM_0 Auxiliary Input 2	I/O		
	IMCR[182]	0001	REQ9	SIUL2	SIUL2 External Interrupt Source 9	I		
	IMCR[214]	0011	SENT_RX[1]	SENT_1	SENT 1 Receiver channel 1	I		
A[11]	MSCR[11]	0000 (Default)	GPIO[11]	SIUL2-GPIO[11]	General Purpose IO A[11]	I/O	120	D10
		0001	SCK	DSPI2	DSPI 2 Serial Clock (output)	I/O		
		0010	A0	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 0	I/O		
		0011	A2	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 2	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[48]	0010	SCK	DSPI2	DSPI 2 Serial Clock (input)	I/O		
	IMCR[88]	0001	A0	FlexPWM_0	FlexPWM_0 Channel A Input 0	I/O		
	IMCR[94]	0001	A2	FlexPWM_0	FlexPWM_0 Channel A Input 2	I/O		
	IMCR[183]	0001	REQ10	SIUL2	SIUL2 External Interrupt Source 10	I		
A[12]	MSCR[12]	0000 (Default)	GPIO[12]	SIUL2-GPIO[12]	General Purpose IO A[12]	I/O	122	D7
		0001	SOUT	DSPI2	DSPI 2 Serial Data Out	O		
		0010	A2	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 2	I/O		
		0011	B2	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 2	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[94]	0010	A2	FlexPWM_0	FlexPWM_0 Channel A Input 2	I/O		
	IMCR[95]	0001	B2	FlexPWM_0	FlexPWM_0 Channel B Input 2	I/O		
	IMCR[184]	0001	REQ11	SIUL2	SIUL2 External Interrupt Source 11	I		
A[13]	MSCR[13]	0000 (Default)	GPIO[13]	SIUL2-GPIO[13]	General Purpose IO A[13]	I/O	136	C5
		0001	—	Reserved	—	—		
		0010	B2	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 2	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[83]	0010	FAULT0	FlexPWM_0	FlexPWM_0 Fault Input 0	I		
	IMCR[95]	0010	B2	FlexPWM_0	FlexPWM_0 Channel B Input 2	I/O		
	IMCR[47]	0001	SIN	DSPI2	DSPI 2 Serial Data Input	I		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[185]	0001	REQ12	SIUL2	SIUL2 External Interrupt Source 12	I		
A[14]	MSCR[14]	0000 (Default)	GPIO[14]	SIUL2-GPIO[14]	General Purpose IO A[14]	I/O	143	A3
		0001	TXD	CAN1	CAN 1 Transmit Pin	O		
		0010	ETC4	eTimer_1	eTimer_1 Input/Output Data Channel 4	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[69]	0001	ETC4	eTimer_1	eTimer_1 Input Data Channel 4	I/O		
	IMCR[186]	0001	REQ13	SIUL2	SIUL2 External Interrupt Source 13	I		
A[15]	MSCR[15]	0000 (Default)	GPIO[15]	SIUL2-GPIO[15]	General Purpose IO A[15]	I/O	144	D3
		0001	—	Reserved	—	—		
		0010	ETC5	eTimer_1	eTimer_1 Input/Output Data Channel 5	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[32]	0001	RXD	CAN0	CAN 0 Receive Pin	I		
	IMCR[33]	0001	RXD	CAN1	CAN 1 Receive Pin	I		
	IMCR[70]	0010	ETC5	eTimer_1	eTimer_1 Input Data Channel 5	I/O		
	IMCR[187]	0001	REQ14	SIUL2	SIUL2 External Interrupt Source 14	I		
B[0]	MSCR[16]	0000 (Default)	GPIO[16]	SIUL2-GPIO[16]	General Purpose IO B[0]	I/O	109	C16
		0001	TXD	CAN0	CAN 0 Transmit Pin	O		
		0010	ETC2	eTimer_1	eTimer_1 Input/Output Data Channel 2	I/O		
		0011	DEBUG0	SSCM	SSCM Debug Output 0	O		
		0100-1111	—	Reserved	—	—		
	IMCR[67]	0001	ETC2	eTimer_1	eTimer_1 Input Data Channel 2	I/O		
	IMCR[188]	0001	REQ15	SIUL2	SIUL2 External Interrupt Source 15	I		
B[1]	MSCR[17]	0000 (Default)	GPIO[17]	SIUL2-GPIO[17]	General Purpose IO B[1]	I/O	110	C14
		0001	—	Reserved	—	—		
		0010	ETC3	eTimer_1	eTimer_1 Input/Output Data Channel 3	I/O		
		0011	DEBUG1	SSCM	SSCM Debug Output 1	O		
		0100-1111	—	Reserved	—	—		
	IMCR[32]	0010	RXD	CAN0	CAN 0 Receive Pin	I		
	IMCR[33]	0010	RXD	CAN1	CAN 1 Receive Pin	I		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[68]	0001	ETC3	eTimer_1	eTimer_1 Input Data Channel 3	I/O		
	IMCR[189]	0001	REQ16	SIUL2	SIUL2 External Interrupt Source 16	I		
B[2]	MSCR[18]	0000 (Default)	GPIO[18]	SIUL2-GPIO[18]	General Purpose IO B[2]	I/O	114	C12
		0001	TXD	LIN0	LINFlexD 0 Transmit Pin	O		
		0010	CS4	DSPI0	DSPI 0 Peripheral Chip Select 4	O		
		0011	DEBUG2	SSCM	SSCM Debug Output 2	O		
		0100-1111	—	Reserved	—	—		
	IMCR[190]	0001	REQ17	SIUL2	SIUL2 External Interrupt Source 17	I		
B[3]	MSCR[19]	0000 (Default)	GPIO[19]	SIUL2-GPIO[19]	General Purpose IO B[3]	I/O	116	B12
		0001	—	Reserved	—	—		
		0010	CS5	DSPI0	DSPI 0 Peripheral Chip Select 5	O		
		0011	DEBUG3	SSCM	SSCM Debug Output 3	O		
		0100-1111	—	Reserved	—	—		
	IMCR[165]	0001	RXD	LIN0	LIN 0 Receive Pin	I		
B[4]	MSCR[20]	0	GPIO[20]	SIUL2-GPIO[20]	General Purpose IO B[4]	I/O	89	G14
		0001 (Default)	TDO	NPC_HNDSHK	NPC_HNDSHK Test Data Out (TDO)	O		
		0010-1111	—	Reserved	—	—		
B[5]	MSCR[21]	0000 (Default)	GPIO[21]	SIUL2-GPIO[21]	JTAGC Test Data In (TDI)[3] General Purpose IO B[5]	I/O	86	J17
		0001	CS7	DSPI0	DSPI 0 Peripheral Chip Select 7	O		
		0010-1111	—	Reserved	—	—		
B[6]	MSCR[22]	0000 (Default)	GPIO[22]	SIUL2-GPIO[22]	General Purpose IO B[6]	I/O	138	B5
		0001	CLK_OUT	MC_CGM	CGM Clock out for off-chip use and observation	O		
		0010	CS2	DSPI2	DSPI 2 Peripheral Chip Select 2	O		
		0011-1111	—	Reserved	—	—		
	IMCR[191]	0001	REQ18	SIUL2	SIUL2 External Interrupt Source 18	I		
B[7]	MSCR[23]	0000 (Default)	GPI[23] ADC0_AN[0]	SIUL2-GPI[23]	General Purpose Input B[7]	I	43	R5

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[165]	0010	RXD	LIN0	LIN 0 Receive Pin	I		
B[8]	MSCR[24]	0	GPI[24][4] ADC0_AN[1]	SIUL2-GPI[24]	General Purpose Input B[8]	I	47	P7
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[64]	0001	ETC5	eTimer_0	eTimer_0 Input Data Channel 5	I/O		
B[9]	MSCR[25]	0000 (Default)	GPI[25][4] ADC0_ADC1_AN[11]	SIUL2-GPI[25]	General Purpose Input B[9]	I	52	U7
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
B[10]	MSCR[26]	0000 (Default)	GPI[26][4] ADC0_ADC1_AN[12]	SIUL2-GPI[26]	General Purpose Input B[10]	I	53	R8
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
B[11]	MSCR[27]	0000 (Default)	GPI[27][4] ADC0_ADC1_AN[13]	SIUL2-GPI[27]	General Purpose Input B[11]	I	54	T8
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
B[12]	MSCR[28]	0000 (Default)	GPI[28][4] ADC0_ADC1_AN[14]	SIUL2-GPI[28]	General Purpose Input B[12]	I	55	U8
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
B[13]	MSCR[29]	0000 (Default)	GPI[29][4] ADC1_AN[0]	SIUL2-GPI[29]	General Purpose Input B[13]	I	60	R10
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[166]	0001	RXD	LIN1	LIN 1 Receive Pin	I		
B[14]	MSCR[30]	0000 (Default)	GPI[30][4] ADC1_AN[1]	SIUL2-GPI[30]	General Purpose Input B[14]	I	64	P11
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[63]	0001	ETC4	eTimer_0	eTimer_0 Input Data Channel 4	I/O		
	IMCR[192]	0001	REQ19	SIUL2	SIUL2 External Interrupt Source 19	I		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
B[15]	MSCR[31]	0000 (Default)	GPI[31][4] ADC1_AN[2]	SIUL2-GPI[31]	General Purpose Input B[15]	I	62	R11
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[193]	0001	REQ20	SIUL2	SIUL2 External Interrupt Source 20	I		
C[0]	MSCR[32]	0000 (Default)	GPI[32][4] ADC1_AN[3]	SIUL2-GPI[32]	General Purpose Input C[0]	I	66	R12
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
C[1]	MSCR[33]	0000 (Default)	GPI[33][4] ADC0_AN[2]	SIUL2-GPI[33]	General Purpose Input C[1]	I	41	T4
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
C[2]	MSCR[34]	0000 (Default)	GPI[34][4] ADC0_AN[3]	SIUL2-GPI[34]	General Purpose Input C[2]	I	45	U5
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
C[4]	MSCR[36]	0000 (Default)	GPIO[36]	SIUL2-GPIO[36]	General Purpose IO C[4]	I/O	11	H3
		0001	CS0	DSPI0	DSPI 0 Peripheral Chip Select 0	I/O		
		0010	X1	FlexPWM_0	FlexPWM_0 Auxiliary Input/ Output 1	I/O		
		0011	DEBUG4	SSCM	SSCM Debug Output 4	O		
		0100-1111	—	Reserved	—	—		
	IMCR[93]	0001	X1	FlexPWM_0	FlexPWM_0 Auxiliary Input 1	I/O		
	IMCR[195]	0001	REQ22	SIUL2	SIUL2 External Interrupt Source 22	I		
C[5]	MSCR[37]	0000 (Default)	GPIO[37]	SIUL2-GPIO[37]	General Purpose IO C[5]	I/O	13	G3
		0001	SCK	DSPI0	DSPI 0 Serial Clock (output)	I/O		
		0010	—	Reserved	—	—		
		0011	DEBUG5	SSCM	SSCM Debug Output 5	O		
		0100-1111	—	Reserved	—	—		
	IMCR[86]	0001	FAULT3	FlexPWM_0	FlexPWM_0 Fault Input 3	I		
	IMCR[196]	0001	REQ23	SIUL2	SIUL2 External Interrupt Source 23	I		
C[6]	MSCR[38]	0000 (Default)	GPIO[38]	SIUL2-GPIO[38]	General Purpose IO C[6]	I/O	142	D4
		0001	SOUT	DSPI0	DSPI 0 Serial Data Out	O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0010	B1	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 1	I/O		
		0011	DEBUG6	SSCM	SSCM Debug Output 6	O		
		0100-1111	—	Reserved	—	—		
	IMCR[92]	0001	B1	FlexPWM_0	FlexPWM_0 Channel B Input 1	I		
	IMCR[197]	0001	REQ24	SIUL2	SIUL2 External Interrupt Source 24	I/O		
C[7]	MSCR[39]	0000 (Default)	GPIO[39]	SIUL2-GPIO[39]	General Purpose IO C[7]	I/O	15	J1
		0001	—	Reserved	—	—		
		0010	A1	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 1	I/O		
		0011	DEBUG7	SSCM	SSCM Debug Output 7	O		
		0100-1111	—	Reserved	—	—		
	IMCR[41]	0001	SIN	DSPI0	DSPI 0 Serial Data Input	I		
	IMCR[91]	0001	A1	FlexPWM_0	FlexPWM_0 Channel A Input 1	I/O		
C[10]	MSCR[42]	0000 (Default)	GPIO[42]	SIUL2-GPIO[42]	General Purpose IO C[10]	I/O	111	B14
		0001	CS2	DSPI2	DSPI 2 Peripheral Chip Select 2	O		
		0010	—	Reserved	—	—		
		0011	A3	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 3	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[84]	0001	FAULT1	FlexPWM_0	FlexPWM_0 Fault Input 1	I		
	IMCR[97]	0010	A3	FlexPWM_0	FlexPWM_0 Channel A Input 3	I/O		
C[11]	MSCR[43]	0000 (Default)	GPIO[43]	SIUL2-GPIO[43]	General Purpose IO C[11]	I/O	80	P16
		0001	ETC4	eTimer_0	eTimer_0 Input/Output Data Channel 4	I/O		
		0010	CS2	DSPI2	DSPI 2 Peripheral Chip Select 2	O		
		0011	TX_ER	ENET_0	Ethernet transmit Data Error	O		
		0100	CS0	DSPI3	DSPI 3 Peripheral Chip Select 0	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[52]	0001	CS0	DSPI3	DSPI 3 Peripheral Chip Select 3	O		
	IMCR[63]	0100	ETC4	eTimer_0	eTimer_0 Input Data Channel 4	I/O		
C[12]	MSCR[44]	0000 (Default)	GPIO[44]	SIUL2-GPIO[44]	General Purpose IO C[12]	I/O	82	M14

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0001	ETC5	eTimer_0	eTimer_0 Input/Output Data Channel 5[5]	I/O		
		0010	CS3	DSPI2	DSPI 2 Peripheral Chip Select 3	O		
		0011	—	LFAST	SIPI/LFAST PLL Phase 0 clock on positive terminal	O		
		0100	CS1	DSPI3	DSPI 3 Peripheral Chip Select 1	O		
		0101-1111	—	Reserved	—	—		
	IMCR[213]	0100	SENT_RX[0]	SENT1	SENT 1 Receiver Channel 0	I		
	IMCR[64]	0011	ETC5	eTimer_0	eTimer_0 Input Data Channel 5	I/O		
C[13]	MSCR[45]	0000 (Default)	GPIO[45]	SIUL2-GPIO[45]	General Purpose IO C[13]	I/O	101	E15
		0001	ETC1	eTimer_1	eTimer_1 Input/Output Data Channel 1	I/O		
		0010-0011	—	Reserved	—	—		
		0100	A0	FlexPWM_1	FlexPWM_1 Channel A Input 0	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[38]	0001	EXT_IN	CTU_0	CTU 0 External Trigger Input	I		
	IMCR[66]	0001	ETC1	eTimer_1	eTimer_1 Input Data Channel 1	I/O		
	IMCR[87]	0001	EXT_SYNC	FlexPWM_0	FlexPWM_0 External Trigger Input	I		
	IMCR[105]	0001	A0	FlexPWM_1	FlexPWM_1 Channel A Input 0	I/O		
C[14]	MSCR[46]	0000 (Default)	GPIO[46]	SIUL2-GPIO[46]	General Purpose IO C[14]	I/O	103	F14
		0001	ETC2	eTimer_1	eTimer_1 Input/Output Data Channel 2	I/O		
		0010	EXT_TGR	CTU_0	CTU0 External Trigger Output	O		
		0011	CS7	DSPI1	DSPI 1 Peripheral Chip Select 7	O		
		0100	B0	FlexPWM_1	FlexPWM_1 Channel B Input/ Output 0	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[67]	0010	ETC2	eTimer_1	eTimer_1 Input Data Channel 2	I/O		
	IMCR[106]	0001	B0	FlexPWM_1	FlexPWM_1 Channel B Input 0	I/O		
C[15]	MSCR[47]	0000 (Default)	GPIO[47]	SIUL2-GPIO[47]	General Purpose IO C[15]	I/O	124	A8
		0001	FR_A_TXEN	FLEXRAY	FlexRay Transmit Enable Channel A	O		
		0010	ETC0	eTimer_1	eTimer_1 Input/Output Data Channel 0	I/O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0011	A1	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 1	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[38]	0010	EXT_IN	CTU_0	CTU 0 External Trigger Input	I		
	IMCR[65]	0010	ETC0	eTimer_1	eTimer_1 Input Data Channel 0	I/O		
	IMCR[87]	0010	EXT_SYNC	FlexPWM_0	FlexPWM_0 External Sync Input	I		
	IMCR[91]	0010	A1	FlexPWM_0	FlexPWM_0 Channel A Input 1	I/O		
D[0]	MSCR[48]	0000 (Default)	GPIO[48]	SIUL2-GPIO[48]	General Purpose IO D[0]	I/O	125	B8
		0001	FR_A_TX	FLEXRAY	FlexRay Transmit Data Channel A	O		
		0010	ETC1	eTimer_1	eTimer_1 Input/Output Data Channel 1	I/O		
		0011	B1	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 1	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[66]	0010	ETC1	eTimer_1	eTimer_1 Input Data Channel 1	I/O		
	IMCR[92]	0010	B1	FlexPWM_0	FlexPWM_0 Channel B Input 1	I/O		
D[1]	MSCR[49]	0000 (Default)	GPIO[49]	SIUL2-GPIO[49]	General Purpose IO D[1]	I/O	3	E3
		0001	—	Reserved	—	—		
		0010	ETC2	eTimer_1	eTimer_1 Input/Output Data Channel 2	I/O		
		0011	EXT_TGR	CTU_0	CTU 0 External Trigger Output	O		
		0100-1111	—	Reserved	—	—		
	IMCR[67]	0011	ETC2	eTimer_1	eTimer_1 Input Data Channel 2	I/O		
	IMCR[136]	0001	FR_A_RX	FLEXRAY	FlexRay Channel A Receive Pin	I		
D[2]	MSCR[50]	0000 (Default)	GPIO[50]	SIUL2-GPIO[50]	General Purpose IO D[2]	I/O	140	B4
		0001	—	Reserved	—	—		
		0010	ETC3	eTimer_1	eTimer_1 Input/Output Data Channel 3	I/O		
		0011	X3	FlexPWM_0	FlexPWM_0 Auxiliary Input/ Output 3	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[68]	0010	ETC3	eTimer_1	eTimer_1 Input Data Channel 3	I/O		
	IMCR[99]	0001	X3	FlexPWM_0	FlexPWM_0 Auxiliary Input 3	I/O		
	IMCR[137]	0001	FR_B_RX	FLEXRAY	FlexRay Channel B Receive Pin	I		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
D[3]	MSCR[51]	0000 (Default)	GPIO[51]	SIUL2-GPIO[51]	General Purpose IO D[3]	I/O	128	A5
		0001	FR_B_TX	FLEXRAY	FlexRay Transmit Data Channel B	O		
		0010	ETC4	eTimer_1	eTimer_1 Input/Output Data Channel 4	I/O		
		0011	A3	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 3	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[69]	0010	ETC4	eTimer_1	eTimer_1 Input Data Channel 4	I/O		
	IMCR[97]	0011	A3	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 3	I/O		
D[4]	MSCR[52]	0000 (Default)	GPIO[52]	SIUL2-GPIO[52]	General Purpose IO D[4]	I/O	129	B7
		0001	FR_B_TXEN	FLEXRAY	FlexRay Transmit Enable Channel B	O		
		0010	ETC5	eTimer_1	eTimer_1 Input/Output Data Channel 5	I/O		
		0011	B3	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 3	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[70]	0011	ETC5	eTimer_1	eTimer_1 Input Data Channel 5	I/O		
	IMCR[98]	0011	B3	FlexPWM_0	FlexPWM_0 Channel B Input 3	I/O		
D[5]	MSCR[53]	0000 (Default)	GPIO[53]	SIUL2-GPIO[53]	General Purpose IO D[5]	I/O	33	M4
		0001	CS3	DSPI0	DSPI 0 Peripheral Chip Select 3	O		
		0010	—	Reserved	—	—		
		0100	SOUT	DSPI3	DSPI 3 Serial Data Out	O		
		0101-1111	—	Reserved	—	—		
	IMCR[85]	0001	FAULT2	FlexPWM_0	FlexPWM_0 Fault Input 2	I		
	IMCR[205]	0001	SENT_RX[0]	SENT0	SENT 0 Receiver channel 0	I		
	IMCR[227]	0001	RX_D1	ENET_0	Ethernet MII/RMII receive data 1	I		
D[6]	MSCR[54]	0000 (Default)	GPIO[54]	SIUL2-GPIO[54]	General Purpose IO D[6]	I/O	34	P3
		0001	CS2	DSPI0	DSPI 0 Peripheral Chip Select 2	O		
		0010	—	Reserved	—	—		
		0011	X3	FlexPWM_0	FlexPWM_0 Auxiliary Input/ Output 3	I/O		
		0100	SCK	DSPI3	DSPI 3 Serial Clock (Output)	I/O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0101-1111	—	Reserved	—	—		
	IMCR[51]	0001	SCK	DSPI3	DSPI 3 Serial Clock (Output)	I/O		
	IMCR[84]	0010	FAULT1	FlexPWM_0	FlexPWM_0 Fault Input 1	I		
	IMCR[99]	0010	X3	FlexPWM_0	FlexPWM_0 Channel X Input 3	I/O		
	IMCR[226]	0001	RX_D0	ENET_0	Ethernet MII/RMII receive data 0	I		
D[7]	MSCR[55]	0000 (Default)	GPIO[55][6] SGEN OUT[7]	SIUL2-GPIO[55]	General Purpose IO D[7]	I/O	37	R4
		0001	CS3	DSPI1	DSPI 1 Peripheral Chip Select 3	O		
		0010	—	Reserved	—	—		
		0011	CS4	DSPI0	DSPI 0 Peripheral Chip Select 4	O		
		0100-1111	—	Reserved	—	—		
	IMCR[50]	0010	SIN	DSPI3	DSPI 3 Serial Data Input	I		
	IMCR[213]	0001	SENT_RX[0]	SENT1	SENT 1 Receiver channel 0	I		
	IMCR[225]	0001	RX_DV	ENET_0	Ethernet Receive data valid	I		
D[8]	MSCR[56]	0000 (Default)	GPIO[56]	SIUL2-GPIO[56]	General Purpose IO D[8]	I/O	32	L4
		0001	CS2	DSPI1	DSPI 1 Peripheral Chip Select 2	O		
		0010	ETC4	eTimer_1	eTimer_1 Input/Output Data Channel 4	I/O		
		0011	CS5	DSPI0	DSPI 0 Peripheral Chip Select 5	O		
		0100-1111	—	Reserved	—	—		
	IMCR[69]	0011	ETC4	eTimer_1	eTimer_1 Input Data Channel 4	I/O		
	IMCR[86]	0010	FAULT3	FlexPWM_0	FlexPWM_0 Fault Input 3	I		
	IMCR[224]	0001	RX_CLK	ENET_0	Ethernet Receive clock	I		
D[9]	MSCR[57]	0000 (Default)	GPIO[57]	SIUL2-GPIO[57]	General Purpose IO D[9]	I/O	26	N3
		0001	X0	FlexPWM_0	FlexPWM_0 Auxiliary Input/ Output 0	I/O		
		0010	TXD	LIN1	LINFlexD 1 Transmit Pin	O		
		0011-1111	—	Reserved	—	—		
D[10]	MSCR[58]	0000 (Default)	GPIO[58]	SIUL2-GPIO[58]	General Purpose IO D[10]	I/O	76	R16
		0001	A0	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 0	I/O		
		0010	—	Reserved	—	—		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0011	TX_D2	ENET_0	Ethernet MII transmit data	O		
		0100	CS0	DSPI3	DSPI 3 Peripheral Chip Select 0	I/O		
		0110-1111	—	Reserved	—	—		
	IMCR[52]	0010	CS0	DSPI3	DSPI 3 Peripheral chip Select 0	I/O		
	IMCR[59]	0001	ETC0	eTimer_0	eTimer_0 Input Data Channel 0	I/O		
	IMCR[88]	0010	A0	FlexPWM_0	FlexPWM_0 Channel A Input 0	I/O		
D[11]	MSCR[59]	0000 (Default)	GPIO[59]	SIUL2-GPIO[59]	General Purpose IO D[11]	I/O	78	P17
		0001	B0	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 0	I/O		
		0010	—	Reserved	—	—		
		0011	CS1	DSPI3	DSPI 3 Peripheral Chip Select 1	O		
		0100	SCK	DSPI3	DSPI 3 Serial Clock (Output)	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[51]	0010	SCK	DSPI3	DSPI 3 Serial Clock (Output)	I/O		
	IMCR[60]	0001	ETC1	eTimer_0	eTimer_0 Input Data Channel 1	I/O		
	IMCR[89]	0010	B0	FlexPWM_0	FlexPWM_0 Channel B Input 0	I/O		
D[12]	MSCR[60]	0000 (Default)	GPIO[60]	SIUL2-GPIO[60]	General Purpose IO D[12]	I/O	99	F15
		0001	X1	FlexPWM_0	FlexPWM_0 Auxiliary Input/ Output 1	I/O		
		0010	CS6	DSPI1	DSPI 1 Peripheral Chip Select 6	O		
		0011	CS2	DSPI3	DSPI 3 Peripheral Chip Select 2	O		
		0100	SOUT	DSPI3	DSPI 1 Serial Data Out	O		
		0101-1111	—	Reserved	—	—		
	IMCR[93]	0010	X1	FlexPWM_0	FlexPWM_0 Channel X Input 1	I/O		
	IMCR[166]	0010	RXD	LIN1	LIN 1 Receive Pin	I		
D[14]	MSCR[62]	0000 (Default)	GPIO[62]	SIUL2-GPIO[62]	General Purpose IO D[14]	I/O	105	E17
		0001	B1	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 1	I/O		
		0010	—	Reserved	—	—		
		0011	CS3	DSPI3	DSPI 3 Peripheral Chip Select 3	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[50]	0011	SIN	DSPI3	DSPI 3 Serial Data Input	I		
	IMCR[62]	0001	ETC3	eTimer_0	eTimer_0 Input Data Channel 3	I/O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[92]	0011	B1	FlexPWM_0	FlexPWM_0 Channel B Input 1	I/O		
E[0]	MSCR[64]	0000 (Default)	GPI[64][4] ADC1_AN[5]/ ADC3_AN[4]	SIUL2-GPI[64]	General Purpose Input E[0]	I	68	T13
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[2]	MSCR[66]	0000 (Default)	GPI[66][4] ADC0_AN[5]	SIUL2-GPI[66]	General Purpose Input E[2]	I	49	U6
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[4]	MSCR[68]	0000 (Default)	GPI[68][4] ADC0_AN[7]	SIUL2-GPI[68]	General Purpose Input E[4]	I	42	U4
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[5]	MSCR[69]	0000 (Default)	GPI[69][4] ADC0_AN[8]	SIUL2-GPI[69]	General Purpose Input E[5]	I	44	T5
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[6]	MSCR[70]	0000 (Default)	GPI[70][4] ADC0_ADC2_AN[4]	SIUL2-GPI[70]	General Purpose Input E[6]	I	46	R6
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[7]	MSCR[71]	0000 (Default)	GPI[71][4] ADC0_AN[6]	SIUL2-GPI[71]	General Purpose Input E[7]	I	48	T6
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[9]	MSCR[73]	0000	GPI[73][4] ADC1_AN[7]/ ADC3_AN[6]	SIUL2-GPI[73]	General Purpose Input E[9]	I	61	U10
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[10]	MSCR[74]	0000 (Default)	GPI[74][4] ADC1_AN[8]/ ADC3_AN[7]	SIUL2-GPI[74]	General Purpose Input E[10]	I	63	T11
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[11]	MSCR[75]	0000 (Default)	GPI[75][4] ADC1_AN[4]/ ADC3_AN[3]	SIUL2-GPI[75]	General Purpose Input E[11]	I	65	U11
		0001	—	Reserved	—	—		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0010-1111	—	Reserved	—	—		
E[12]	MSCR[76]	0000 (Default)	GPI[76][4] ADC1_AN[6]/ ADC3_AN[5]	SIUL2-GPI[76]	General Purpose Input E[12]	I	67	T12
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
E[13]	MSCR[77]	0000 (Default)	GPIO[77]	SIUL2-GPIO[77]	General Purpose IO E[13]	I/O	117	A11
		0001	ETC5	eTimer_0	eTimer_0 Input/Output Data Channel 5	I/O		
		0010	CS3	DSPI2	DSPI 2 Peripheral Chip Select 3	O		
		0011	CS4	DSPI1	DSPI 1 Peripheral Chip Select 4	O		
		0100	SCK	DSPI3	DSPI 3 Serial Clock (Output)	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[51]	0011	SCK	DSPI3	DSPI 3 Serial Clock (Output)	I/O		
	IMCR[198]	0001	REQ25	SIUL2	SIUL2 External Interrupt Source 25	I		
	IMCR[64]	0100	ETC5	eTimer_0	eTimer_0 Input Data Channel	I/O		
E[14]	MSCR[78]	0000 (Default)	GPIO[78]	SIUL2-GPIO[78]	General Purpose IO E[14]	I/O	119	B10
		0001	ETC5	eTimer_1	eTimer_1 Input/Output Data Channel 5	I/O		
		0010	SOUT	DSPI3	DSPI 3 Serial Data Out	O		
		0011	CS5	DSPI1	DSPI 1 Peripheral Chip Select 5	O		
		0100	B2	FlexPWM_1	FlexPWM_1 Channel B Input/ Output 2	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[70]	0100	ETC5	eTimer_1	eTimer_1 Input Data Channel 5	I/O		
	IMCR[113]	0001	B2	FlexPWM_1	FlexPWM_1 Channel B Input 2	I/O		
	IMCR[199]	0001	REQ26	SIUL2	SIUL2 External Interrupt Source 26	I		
E[15]	MSCR[79]	0000 (Default)	GPIO[79]	SIUL2-GPIO[79]	General Purpose IO E[15]	I/O	121	C8
		0001	CS1	DSPI0	DSPI 0 Peripheral Chip Select 1	O		
		0010	—	Reserved	—	—		
		0011	TIMER1	ENET_0	Ethernet TIMER Outputs (Output Compare Events)	O		
		0100-1111	—	Reserved	—	—		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[50]	0100	SIN	DSPI3	DSPI 3 Serial Data Input	I		
	IMCR[200]	0001	REQ27	SIUL2	SIUL2 External Interrupt Source 27	I		
F[0]	MSCR[80]	0000 (Default)	GPIO[80]	SIUL2-GPIO[80]	General Purpose IO F[0]	I/O	133	B6
		0001	A1	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 1	I/O		
		0010	CS3	DSPI3	DSPI 3 Peripheral Chip Select 3	I/O		
		0011	MDC	ENET_0	Ethernet MDIO clock output	O		
		0100-1111	—	Reserved	—	—		
	IMCR[61]	0001	ETC2	eTimer_0	eTimer_0 Input Data Channel 2	I/O		
	IMCR[91]	0011	A1	FlexPWM_0	FlexPWM_0 Channel A Input 1	I/O		
	IMCR[201]	0001	REQ28	SIUL2	SIUL2 External Interrupt Source 28	I		
F[3]	MSCR[83]	0000 (Default)	GPIO[83]	SIUL2-GPIO[83]	General Purpose IO F[3]	I/O	139	B3
		0001	CS6	DSPI0	DSPI 0 Peripheral Chip Select 6	O		
		0010	—	Reserved	—	—		
		0011	CS2	DSPI3	DSPI 3 Peripheral Chip Select 2	O		
		0100	TIMER2	ENET_0	Ethernet TIMER Outputs 2 (Output Compare Events)	I/O		
		0101-1111	—	Reserved	—	—		
F[4]	MSCR[84]	0000 (Default)	GPIO[84]	SIUL2-GPIO[84]	General Purpose IO F[4]	I/O	4	E1
		0001	—	Reserved	—	I/O		
		0010	MDO[3]	NPC_WRAPPER	Nexus - Message Data Out Pin 3	O		
		0011	CS1	DSPI3	DSPI 3 Peripheral Chip Select 1	O		
		0100-1111	—	Reserved	—	—		
F[5]	MSCR[85]	0000 (Default)	GPIO[85]	SIUL2-GPIO[85]	General Purpose IO F[5]	I/O	5	F1
		0001	—	Reserved	—	I/O		
		0010	MDO[2]	NPC_WRAPPER	Nexus Message Data Out Pin 2	O		
		0011	CS0	DSPI3	DSPI 3 Peripheral Chip Select 0	I/O		
		0100-1111	—	Reserved	—	—		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[52]	0011	CS0	DSPI3	DSPI 3 Peripheral Chip Select 0	I/O		
F[6]	MSCR[86]	0000 (Default)	GPIO[86]	SIUL2-GPIO[86]	General Purpose IO F[6]	I/O	8	E2
		0001	—	Reserved	—	I/O		
		0010	MDO[1]	NPC_WRAPPER	Nexus Message Data Out Pin 1	O		
		0011-1111	—	Reserved	—	—		
F[7]	MSCR[87]	0000 (Default)	GPIO[87]	SIUL2-GPIO[87]	General Purpose IO F[7]	I/O	19	J4
		0001	—	Reserved	—	I/O		
		0010	MCKO	NPC_WRAPPER	Nexus Message Clock Out for development tools	O		
		0011-1111	—	Reserved	—	—		
F[8]	MSCR[88]	0000 (Default)	GPIO[88]	SIUL2-GPIO[88]	General Purpose IO F[8]	I/O	20	J3
		0001	—	Reserved	—	I/O		
		0010	MSEO_B[1]	NPC_WRAPPER	Nexus Message Start/End Out Pin 1	O		
		0011-1111	—	Reserved	—	—		
F[9]	MSCR[89]	0000 (Default)	GPIO[89]	SIUL2-GPIO[89]	General Purpose IO F[9]	I/O	23	K3
		0001	—	Reserved	—	I/O		
		0010	MSEO_B[0]	NPC_WRAPPER	Nexus Message Start/End Out Pin 0	O		
		0011-1111	—	Reserved	—	—		
F[10]	MSCR[90]	0000 (Default)	GPIO[90]	SIUL2-GPIO[90]	General Purpose IO F[10]	I/O	24	K2
		0001	—	Reserved	—	—		
		0010	EVTO_B	NPC_WRAPPER	Nexus Event Out Pin	O		
		0011-1111	—	Reserved	—	—		
F[11]	MSCR[91]	0000 (Default)	GPIO[91]	SIUL2-GPIO[91]	General Purpose IO F[11]	I/O	25	L2
		0001	—	Reserved	—	—		
		0010	EVTI_IN	NPC_WRAPPER	Nexus Event In Pin	I		
		0011-1111	—	Reserved	—	—		
F[12]	MSCR[92]	0000 (Default)	GPIO[92]	SIUL2-GPIO[92]	General Purpose IO F[12]	I/O	106	D17
		0001	ETC3	eTimer_1	eTimer_1 Input/Output Data Channel 3	I/O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0010-0011	—	Reserved	—	—		
		0100	A1	FlexPWM_1	FlexPWM_1 Channel A Input 1	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[68]	0011	ETC3	eTimer_1	eTimer_1 Input Data Channel 3	I/O		
	IMCR[109]	0001	A1	FlexPWM_1	FlexPWM_1 Channel A Input 1	I/O		
	IMCR[203]	0001	REQ30	SIUL2	SIUL2 External Interrupt Source 30	I		
F[13]	MSCR[93]	0000 (Default)	GPIO[93]	SIUL2-GPIO[93]	General Purpose IO F[13]	I/O	112	A15
		0001	ETC4	eTimer_1	eTimer_1 Input/Output Data Channel 4	I/O		
		0010-0011	—	Reserved	—	—		
		0100	B1	FlexPWM_1	FlexPWM_1 Channel B Input/ Output 1	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[69]	0100	ETC4	eTimer_1	eTimer_1 Input Data Channel 4	I/O		
	IMCR[110]	0001	B1	FlexPWM_1	FlexPWM_1 Channel B Input 1	I/O		
	IMCR[204]	0001	REQ31	SIUL2	SIUL2 External Interrupt Source 31	I		
F[14]	MSCR[94]	0000 (Default)	GPIO[94]	SIUL2-GPIO[94]	General Purpose IO F[14]	I/O	115	D12
		0001	TXD	LIN1	LINFlexD 1 Transmit Pin	O		
		0010	TXD	CAN2	CAN 2 Transmit Pin	O		
		0011-1111	—	Reserved	—	—		
F[15]	MSCR[95]	0000 (Default)	GPIO[95]	SIUL2-GPIO[95]	General Purpose IO F[15]	I/O	113	A13
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[166]	0011	RXD	LIN1	LIN1 RXD	I		
	IMCR[34]	0001	RXD	CAN2	CAN2 RXD	I		
G[2]	MSCR[98]	0000 (Default)	GPIO[98]	SIUL2-GPIO[98]	General Purpose IO G[2]	I/O	102	F17
		0001	X2	FlexPWM_0	FlexPWM_0 Auxiliary Input/ Output 2	I/O		
		0010	CS1	DSPI1	DSPI 1 Peripheral Chip Select 1	O		
		0011-1111	—	Reserved	—	—		
	IMCR[96]	0010	X2	FlexPWM_0	FlexPWM_0 Auxiliary Input 2	I/O		
G[3]	MSCR[99]	0000 (Default)	GPIO[99]	SIUL2-GPIO[99]	General Purpose IO G[3]	I/O	104	E16

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0001	A2	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 2	I/O		
		0010-1111	—	Reserved	—	—		
	IMCR[63]	0010	ETC4	eTimer_0	eTimer_0 Input Data Channel 4	I/O		
	IMCR[94]	0011	A2	FlexPWM_0	FlexPWM_0 Channel A Input 2	I/O		
G[4]	MSCR[100]	0000 (Default)	GPIO[100]	SIUL2-GPIO[100]	General Purpose IO G[4]	I/O	100	F16
		0001	B2	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 2	I/O		
		0010-1111	—	Reserved	—	—		
	IMCR[64]	0010	ETC5	eTimer_0	eTimer_0 Input Data Channel 5	I/O		
	IMCR[95]	0011	B2	FlexPWM_0	FlexPWM_0 Channel B Input 2	I/O		
G[5]	MSCR[101]	0000 (Default)	GPIO[101]	SIUL2-GPIO[101]	General Purpose IO G[5]	I/O	85	M17
		0001	X3	FlexPWM_0	FlexPWM_0 Auxiliary Input/ Output 3	I/O		
		0010	CS3	DSPI2	DSPI 2 Peripheral Chip Select 3	O		
		0011	TX_EN	ENET_0	Ethernet Transmit Data Valid	O		
		0100-1111	—	Reserved	—	—		
	IMCR[99]	0011	X3	FlexPWM_0	FlexPWM_0 Auxiliary Input 3	I/O		
G[6]	MSCR[102]	0000 (Default)	GPIO[102]	SIUL2-GPIO[102]	General Purpose IO G[6]	I/O	98	G17
		0001	A3	FlexPWM_0	FlexPWM_0 Channel A Input/ Output 3	I/O		
		0010-1111	—	Reserved	—	—		
	IMCR[97]	0100	A3	FlexPWM_0	FlexPWM_0 Channel A Input 3	I/O		
G[7]	MSCR[103]	0000 (Default)	GPIO[103]	SIUL2-GPIO[103]	General Purpose IO G[7][8]	I/O	83	M16
		0001	B3	FlexPWM_0	FlexPWM_0 Channel B Input/ Output 3	I/O		
		0010	—	Reserved	—	—		
		0011	—	LFAST	LVDS receive positive terminal	I		
		0100-1111	—	Reserved	—	—		
	IMCR[98]	0100	B3	FlexPWM_0	FlexPWM_0 Channel B Input 3	I/O		
G[8]	MSCR[104]	0000 (Default)	GPIO[104]	SIUL2-GPIO[104]	General Purpose IO G[8]	I/O	81	N14
		0001	FR_DBG[0]	FLEXRAY	FlexRay Debug Strobe Signal 0	O		
		0010	CS1	DSPI0	DSPI 0 Peripheral Chip Select 1	O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0011	RMII_CLK	ENET_0	Ethernet RMII Clock (used in MII to RMII Gaskets)	O		
		0100-1111	—	Reserved	—	—		
	IMCR[83]	0011	FAULT0	FlexPWM_0	FlexPWM_0 Fault Input 0	I		
	IMCR[194]	0001	REQ21	SIUL2	SIUL2 External Interrupt Source 21	I		
	IMCR[205]	0011	SENT_RX[0]	SENT_0	SENT 0 Receiver channel 0	I		
	IMCR[233]	0001	TX_CLK	ENET_0	Ethernet Transmit Clock	I		
G[9]	MSCR[105]	0000 (Default)	GPIO[105]	SIUL2-GPIO[105]	General Purpose IO G[9]	I/O	79	P14
		0001	FR_DBG[1]	FLEXRAY	FlexRay Debug Strobe Signal 1	O		
		0010	CS1	DSPI1	DSPI 1 Peripheral Chip Select 1	O		
		0011	TX_D0	ENET_0	Ethernet MII/RMII transmit data 0	O		
		0100-1111	—	Reserved	—	—		
	IMCR[84]	0011	FAULT1	FlexPWM_0	FlexPWM_0 Fault Input 1	I		
	IMCR[202]	0001	REQ29	SIUL2	SIUL2 External Interrupt Source 29	I		
	IMCR[213]	0011	SENT_RX[0]	SENT_1	SENT 1 Receiver channel 0	I		
G[10]	MSCR[106]	0000 (Default)	GPIO[106]	SIUL2-GPIO[106]	General Purpose IO G[10]	I/O	77	R17
		0001	FR_DBG[2]	FLEXRAY	FlexRay Debug Strobe Signal 2	O		
		0010	CS3	DSPI2	DSPI 2 Peripheral Chip Select 3	O		
		0011	TX_D1	ENET_0	Ethernet MII/RMII transmit data 1	O		
		0100-1111	—	Reserved	—	—		
	IMCR[85]	0010	FAULT2	FlexPWM_0	FlexPWM_0 Fault Input 2	I		
	IMCR[206]	0100	SENT_RX[1]	SENT_0	SENT 0 Receiver channel 1	I		
G[11]	MSCR[107]	0000 (Default)	GPIO[107]	SIUL2-GPIO[107]	General Purpose IO G[11]	I/O	75	T15
		0001	FR_DBG[3]	FLEXRAY	FlexRay Debug Strobe Signal 3	O		
		0010	—	Reserved	—	—		
		0011	TX_D3	ENET_0	Ethernet MII/RMII transmit data 3	O		
		0100-1111	—	Reserved	—	—		
	IMCR[86]	0011	FAULT3	FlexPWM_0	FlexPWM_0 Fault Input 3	I		
	IMCR[214]	0100	SENT_RX[1]	SENT_1	SENT 1 Receiver channel 1	I		
H[4]	MSCR[116]	0000 (Default)	GPIO[116]	SIUL2-GPIO[116]	General Purpose IO H[4]	I/O		F4

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0001	X0	FlexPWM_1	FlexPWM_1 Auxiliary Input/ Output 0	I/O		
		0010	ETC0	eTimer_2	eTimer_2 Input/Output Data Channel 0	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[71]	0001	ETC0	eTimer_2	eTimer_2 Input Data Channel 0	I/O		
	IMCR[231]	0001	CRS	ENET_0	Ethernet MII Carrier Sense	I		
H[5]	MSCR[117]	0000 (Default)	GPIO[117]	SIUL2-GPIO[117]	General Purpose IO H[5]	I/O		F3
		0001	A0	FlexPWM_1	FlexPWM_1 Channel A Input/ Output 0	I/O		
		0010	—	Reserved	—	—		
		0011	CS4	DSPI0	DSPI 0 Peripheral Chip Select 4	O		
		0100-1111	—	Reserved	—	—		
	IMCR[105]	0010	A0	FlexPWM_1	FlexPWM_1 Channel A Input 0	I/O		
	IMCR[230]	0001	COL	ENET_0	Ethernet MII Collision	I		
H[6]	MSCR[118]	0000 (Default)	GPIO[118]	SIUL2-GPIO[118]	General Purpose IO H[6]	I/O		C13
		0001	B0	FlexPWM_1	FlexPWM_1 Channel B Input/ Output 0	I/O		
		0010	—	Reserved	—	—		
		0011	CS5	DSPI0	DSPI 0 Peripheral Chip Select 5	O		
		0100-1111	—	Reserved	—	—		
	IMCR[106]	0010	B0	FlexPWM_1	FlexPWM_1 Channel B Input 0	I/O		
H[7]	MSCR[119]	0000 (Default)	GPIO[119]	SIUL2-GPIO[119]	General Purpose IO H[7]	I/O		F2
		0001	X1	FlexPWM_1	FlexPWM_1 Auxiliary Input/ Output 1	I/O		
		0010	ETC1	eTimer_2	eTimer_2 Input/Output Data Channel 1	I/O		
		0011	MDIO	ENET_0	Ethernet MDIO input/output data	I/O		
		0100-1111	—	Reserved	—	—		
	IMCR[72]	0001	ETC1	eTimer_2	eTimer_2 Input Data Channel 1	I/O		
H[8]	MSCR[120]	0000 (Default)	GPIO[120]	SIUL2-GPIO[120]	General Purpose IO H[8]	I/O		L1
		0001	A1	FlexPWM_1	FlexPWM_1 Channel A Input/ Output 1	I/O		
		0010	—	Reserved	—	—		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0011	CS6	DSPI0	DSPI 0 Peripheral Chip Select 6	O		
		0100-1111	—	Reserved	—	—		
	IMCR[109]	0010	A1	FlexPWM_1	FlexPWM_1 Channel A Input 1	I/O		
	IMCR[228]	0001	RX_D2	ENET_0	Ethernet MII Receive Data 2	I		
H[9]	MSCR[121]	0000 (Default)	GPIO[121]	SIUL2-GPIO[121]	General Purpose IO H[9]	I/O		B13
		0001	B1	FlexPWM_1	FlexPWM_1 Channel B Input/ Output 1	I/O		
		0010	—	Reserved	—	—		
		0011	CS7	DSPI0	DSPI 0 Peripheral Chip Select 7	O		
		0100-1111	—	Reserved	—	—		
	IMCR[110]	0010	B1	FlexPWM_1	FlexPWM_1 Channel B Input 1	I/O		
H[10]	MSCR[122]	0000 (Default)	GPIO[122]	SIUL2-GPIO[122]	General Purpose IO H[10]	I/O		C7
		0001	X2	FlexPWM_1	FlexPWM_1 Auxiliary Input/ Output 2	I/O		
		0010	ETC2	eTimer_2	eTimer_2 Input/Output Data Channel 2	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[73]	0010	ETC2	eTimer_2	eTimer_2 Input Data Channel 2	I/O		
H[11]	MSCR[123]	0000 (Default)	GPIO[123]	SIUL2-GPIO[123]	General Purpose IO H[11]	I/O		C9
		0001	A2	FlexPWM_1	FlexPWM_1 Channel A Input/ Output 2	I/O		
		0010-1111	—	Reserved	—	—		
	IMCR[112]	0010	A2	FlexPWM_1	FlexPWM_1 Channel A Input 2	I/O		
H[12]	MSCR[124]	0000 (Default)	GPIO[124]	SIUL2-GPIO[124]	General Purpose IO H[12]	I/O		A7
		0001	B2	FlexPWM_1	FlexPWM_1 Channel B Input/ Output 2	I/O		
		0010-1111	—	Reserved	—	—		
	IMCR[113]	0010	B2	FlexPWM_1	FlexPWM_1 Channel B Input 2	I/O		
H[13]	MSCR[125]	0000 (Default)	GPIO[125]	SIUL2-GPIO[125]	General Purpose IO H[13]	I/O		A14
		0001	X3	FlexPWM_1	FlexPWM_1 Auxiliary Input/ Output 3	I/O		
		0010	ETC3	eTimer_2	eTimer_2 Input/Output Data Channel 3	I/O		
		0011-1111	—	Reserved	—	—		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
	IMCR[74]	0010	ETC3	eTimer_2	eTimer_2 Input Data Channel 3	I/O		
H[14]	MSCR[126]	0000 (Default)	GPIO[126]	SIUL2-GPIO[126]	General Purpose IO H[14]	I/O		P13
		0001	A3	FlexPWM_1	FlexPWM_1 Channel A Input/ Output 3	I/O		
		0010	ETC4	eTimer_2	eTimer_2 Input/Output Data Channel 4	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[75]	0010	ETC4	eTimer_2	eTimer_2 Input Data Channel 4	I/O		
H[15]	MSCR[127]	0000 (Default)	GPIO[127]	SIUL2-GPIO[127]	General Purpose IO H[15]	I/O		C17
		0001	B3	FlexPWM_1	FlexPWM_1 Channel B Input/ Output 3	I/O		
		0010	ETC5	eTimer_2	eTimer_2 Input/Output Data Channel 5	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[76]	0010	ETC5	eTimer_2	eTimer_2 Input Data Channel 5	I/O		
I[0]	MSCR[128]	0000 (Default)	GPIO[128]	SIUL2-GPIO[128]	General Purpose IO I[0]	I/O		C6
		0001	ETC0	eTimer_2	eTimer_2 Input/Output Data Channel 0	I/O		
		0010	CS4	DSPI0	DSPI 0 Peripheral Chip Select 4	O		
		0011-1111	—	Reserved	—	—		
	IMCR[71]	0010	ETC0	eTimer_2	eTimer_2 Input Data Channel 0	I/O		
	IMCR[100]	0001	FAULT0	FlexPWM_1	FlexPWM_1 Fault Input 0	I		
I[1]	MSCR[129]	0000 (Default)	GPIO[129]	SIUL2-GPIO[129]	General Purpose IO I[1]	I/O		T3
		0001	ETC1	eTimer_2	eTimer_2 Input/Output Data Channel 1	I/O		
		0010	CS5	DSPI0	DSPI 0 Peripheral Chip Select 5	O		
		0011-1111	—	Reserved	—	—		
	IMCR[72]	0010	ETC1	eTimer_2	eTimer_2 Input Data Channel 1	I/O		
	IMCR[101]	0001	FAULT1	FlexPWM_1	FlexPWM_1 Fault Input 1	I		
	IMCR[232]	0001	RX_ER	ENET_0	Ethernet Receive Data Error	I		
I[2]	MSCR[130]	0000 (Default)	GPIO[130]	SIUL2-GPIO[130]	General Purpose IO I[2]	I/O		D11
		0001	ETC2	eTimer_2	eTimer_2 Input/Output Data Channel 2	I/O		
		0010	CS6	DSPI0	DSPI 0 Peripheral Chip Select 6	O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0011-1111	—	Reserved	—	—		
	IMCR[73]	0011	ETC2	eTimer_2	eTimer_2 Input Data Channel 2	I/O		
	IMCR[102]	0001	FAULT2	FlexPWM_1	FlexPWM_1 Fault Input 2	I		
I[3]	MSCR[131]	0000 (Default)	GPIO[131]	SIUL2-GPIO[131]	General Purpose IO I[3]	I/O		A10
		0001	ETC3	eTimer_2	eTimer_2 Input/Output Data Channel 3	I/O		
		0010	CS7	DSPI0	DSPI 0 Peripheral Chip Select 7	O		
		0011	EXT_TGR	CTU_0	CTU0 External Trigger Output	O		
		0100	TIMER0	ENET_0	Ethernet TIMER Outputs 0 (Output Compare Events)	I/O		
		0101-1111	—	Reserved	—	—		
	IMCR[74]	0011	ETC3	eTimer_2	eTimer_2 Input Data Channel 3	I/O		
	IMCR[103]	0001	FAULT3	FlexPWM_1	FlexPWM_1 Fault Input 3	I		
RDY_B/I[4]	MSCR[132]	0000 (Default)	GPIO[132]	SIUL2-GPIO[132]	General Purpose IO I[4]	I/O		J2
		0001	—	Reserved	—	—		
		0010	NEX_RDY_B	NPC_WRAPPER	Nexus data ready for transfer (RDY_B)	O		
		0011-1111	—	Reserved	—	—		
I[5]	MSCR[133]	0000 (Default)	GPIO[133]	SIUL2-GPIO[133]	General Purpose IO I[5][9]	I/O		N15
		0001	TXD	CAN2	CAN 2 Transmit Pin	O		
		0010	—	Reserved	—	—		
		0011	—	LFAST	SIPI/LFAST PLL Phase 0 clock on negative terminal	O		
		0100-1111	—	Reserved	—	—		
I[6]	MSCR[134]	0000 (Default)	GPIO[134]	SIUL2-GPIO[134]	General Purpose IO I[6][10]	I/O		M15
		0001	—	Reserved	—	—		
		0010	—	Reserved	—	—		
		0011	—	LFAST	LVDS receive negative terminal	I		
		0100-1111	—	Reserved	—	—		
	IMCR[34]	0010	RXD	CAN2	CAN 2 Receive Pin	I		
I[7]	MSCR[135]	0000 (Default)	GPIO[135]	SIUL2-GPIO[135]	General Purpose IO I[7]	I/O		D2
		0001	LFAST_REF_CLK	MC_CGM	SIPI LFAST reference clock	—		
		0010-1111	—	Reserved	—	—		
	IMCR[205]	0010	SENT_RX[0]	SENT0	SENT 0 Receiver channel 0	I		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
I[8]	MSCR[136]	0000 (Default)	GPIO[136]	SIUL2-GPIO[136]	General Purpose IO I[8]	I/O		K4
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[213]	0010	SENT_RX[0]	SENT1	SENT 1 Receiver channel 0	I		
I[9]	MSCR[137]	0000 (Default)	GPIO[137]	SIUL2-GPIO[137]	General Purpose IO I[9]	I/O		L3
		0001	ETC4	eTimer_2	eTimer_2 Input/Output Data Channel 4	I/O		
		0010-1111	—	Reserved	—	—		
	IMCR[75]	0011	ETC4	eTimer_2	eTimer_2 Input Data Channel 4	I/O		
I[10]	MSCR[138]	0000 (Default)	GPIO[138]	SIUL2-GPIO[138]	General Purpose IO I[10]	I/O		M3
		0001	ETC5	eTimer_2	eTimer_2 Input/Output Data Channel 5	I/O		
		0010-1111	—	Reserved	—	—		
	IMCR[76]	0011	ETC5	eTimer_2	eTimer_2 Input Data Channel 5	I/O		
I[11]	MSCR[139]	0000 (Default)	GPIO[139]	SIUL2-GPIO[139]	General Purpose IO I[11]	I/O		U3
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[206]	0001	SENT_RX[1]	SENT0	SENT 0 Receiver channel 1	I		
I[12]	MSCR[140]	0000 (Default)	GPIO[140]	SIUL2-GPIO[140]	General Purpose IO I[12]	I/O		P5
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[214]	0001	SENT_RX[1]	SENT1	SENT 1 Receiver channel 1	I		
I[13]	MSCR[141]	0000	GPIO[141]	SIUL2-GPIO[141]	General Purpose IO I[13]	I/O		P6
		0001	EXT_TGR	CTU_1	CTU1 External Trigger Output	I/O		
		0010-1111	—	Reserved	—	—		
I[14]	MSCR[142]	0000 (Default)	GPIO[142]	SIUL2-GPIO[142]	General Purpose IO I[14]	I/O		C10
		0001	CS0	DSPI3	DSPI 3 Peripheral Chip Select 0	I/O		
		0010-1111	—	Reserved	—	—		
	IMCR[52]	0100	CS0	DSPI3	DSPI 3 Peripheral Chip Select 0	I/O		
I[15]	MSCR[143]	0000 (Default)	GPIO[143]	SIUL2-GPIO[143]	General Purpose IO I[15]	I/O		C1
		0001	SCK	DSPI3	DSPI 3 Serial Clock (output)	I/O		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
		0010-1111	—	Reserved	—	—		
	IMCR[51]	0100	SCK	DSPI3	DSPI 3 Peripheral Serial Clock (Output)	I/O		
J[0]	MSCR[144]	0000 (Default)	GPIO[144]	SIUL2-GPIO[144]	General Purpose IO J[0]	I/O		C2
		0001	SOUT	DSPI3	DSPI 3 Serial Data Out	O		
		0010-1111	—	Reserved	—	—		
J[1]	MSCR[145]	0000 (Default)	GPIO[145]	SIUL2-GPIO[145]	General Purpose IO J[1]	I/O		A12
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[50]	0001	SIN	DSPI3	DSPI 3 Serial Data Input	I		
J[2]	MSCR[146]	0000 (Default)	GPIO[146]	SIUL2-GPIO[146]	General Purpose IO J[2]	I/O		C11
		0001	CS1	DSPI3	DSPI 3 Peripheral Chip Select 1	O		
		0010-1111	—	Reserved	—	—		
J[3]	MSCR[147]	0000 (Default)	GPIO[147]	SIUL2-GPIO[147]	General Purpose IO J[3]	I/O		B15
		0001	CS2	DSPI3	DSPI 3 Peripheral Chip Select 2	O		
		0010-1111	—	Reserved	—	—		
J[4]	MSCR[148]	0000 (Default)	GPIO[148]	SIUL2-GPIO[148]	General Purpose IO J[4]	I/O		D13
		0001	CS3	DSPI3	DSPI 3 Peripheral Chip Select 3	O		
		0010-1111	—	Reserved	—	—		
	IMCR[39]	0001	EXT_IN	CTU_1	CTU 1 External Trigger Input	I		
J[5]	MSCR[149]	0000 (Default)	GPI[149][4] ADC2_ADC3_AN[0]	SIUL2-GPI[149]	General Purpose Input J[5]	I		P8
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[206]	0010	SENT_RX[1]	SENT0	SENT 0 Receiver channel 1	I		
J[6]	MSCR[150]	0000 (Default)	GPI[150][4] ADC2_ADC3_AN[1]	SIUL2-GPI[150]	General Purpose Input J[6]	I		P9
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
	IMCR[214]	0010	SENT_RX[1]	SENT1	SENT 1 Receiver channel 1	I		

Table continues on the next page...

Table 4-7. Pin muxing (continued)

Port Pin	SIUL2 MSCR/ IMCR Number	MSCR/ IMCR SSS Value[1]	Signal	Module	Short Signal Description	Dir	LQFP144	BGA257
J[7]	MSCR[151]	0000 (Default)	GPI[151][4] ADC2_ADC3_AN[2]	SIUL2-GPI[151]	General Purpose Input J[7]	I		P10
		0001	—	Reserved	—	—		
		0010-1111	—	Reserved	—	—		
J[8]	MSCR[152]	0000 (Default)	GPIO[152]	SIUL2-GPIO[152]	General Purpose IO J[8]	I/O	95	G16
		0001	ETC4	eTimer_2	eTimer_2 Input/Output Data Channel 4	I/O		
		0010	ETC2	eTimer_2	eTimer_2 Input/Output Data Channel 2	I/O		
		0011-1111	—	Reserved	—	—		
	IMCR[34]	0011	RXD	CAN2	CAN 2 Receive Pin	I		
	IMCR[73]	0100	ETC2	eTimer_2	eTimer_2 Input Data Channel 2	I/O		
	IMCR[75]	0100	ETC4	eTimer_2	eTimer_2 Input Data Channel 4	I/O		
J[9]	MSCR[153]	0000 (Default)	GPIO[153]	SIUL2-GPIO[153]	General Purpose IO J[9]	I/O	16	K1
		0001	ETC5	eTimer_2	eTimer_2 Input/Output Data Channel 5	I/O		
		0010	NEX_RDY_B	NPC	Nexus data ready for transfer (RDY_B)	O		
		0011-1111	—	Reserved	—	—		
	IMCR[39]	0010	EXT_IN	CTU_1	CTU_1 External Trigger Input	I		
	IMCR[76]	0100	ETC5	eTimer_2	eTimer_2 Input Data Channel 5	I/O		
	IMCR[229]	0001	RX_D3	ENET_0	Ethernet MII Receive Data 3	I		
NMI_B	MSCR[154]	0000 (Default)	NMI_B	Core	Non-Maskable Interrupt	I	1	E4

1. Selecting an alternative function with a "Reserved" source function causes the pin to enter a null state (input buffer and output buffer enables are both 0).
2. **(Default) = ALT mode configuration after reset.**
3. Changing the B[5] configuration during debug might affect the availability of TDI.
4. ADC analog input: Program corresponding MSCR APC bit and enable ADC to switch on the analog input path. See Table 1 for details.
5. Shared with SIPI LFAST transmit pad SIPI_TXP. Alternative modes and GPIO must be disabled (OBE=0, IBE=0) if port is used for SIPI LFAST.
6. To operate D[7] as GPIO, disable the Sine Wave Generator (SGEN) and the peripheral bus clock of the SGEN: Program the MC_ME_PCTL239 register to select an MC_ME_RUN_PCn (or MC_ME_LP_PCn) configuration where the field for the desired mode is 0.
7. SGEN output if SGEN is enabled. See Table 1 for details.
8. Shared with SIPI LFAST receive pad SIPI_RXP. Alternative modes and GPIO must be disabled (OBE=0, IBE=0) if port is used for SIPI LFAST.
9. Shared with SIPI LFAST receive pad SIPI_TXN. Alternative modes and GPIO must be disabled (OBE=0, IBE=0) if port is used for SIPI LFAST.
10. Shared with SIPI LFAST receive pad SIPI_RXN. Alternative modes and GPIO must be disabled (OBE=0, IBE=0) if port is used for SIPI LFAST.

The following table list ports that are not implemented. The corresponding control and data registers are not implemented.

Table 4-8. Ports - Not Implemented

Port Name	Port Index
C	3,8,9
D	13,15
E	1,3,8
F	1,2
G	0,1,[12:15]
H	[0:3]
J	[10:15]

Any attempt to access unimplemented MSCRs generates a bus error. The read value from unimplemented ports must be masked in case of parallel port accesses.

4.3.6 Peripheral input muxing

The following table describes the peripheral muxing capabilities of the device. See Table 4-17 for IMCR register addresses.

Table 4-9. Peripheral muxing

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
FlexCAN_0	RXD	IMCR[32]	0000 (Default)[1]	—	Disable
			0001	I/O-Pad	A[15]
			0010	I/O-Pad	B[1]
			0011-1111	—	Reserved[2]
FlexCAN_1	RXD	IMCR[33]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[15]
			0010	I/O-Pad	B[1]
			0011-1111	—	Reserved
FlexCAN_2	RXD	IMCR[34]	0000 (Default)	—	Disable
			0001	I/O-Pad	F[15]
			0010	I/O-Pad	I[6]
			0011	I/O-Pad	J[8]
			0100-1111	—	Reserved
CTU_0	EXT_IN	IMCR[38]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[13]
			0010	I/O-Pad	C[15]

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0011-1111	—	Reserved
CTU_1	EXT_IN	IMCR[39]	0000 (Default)	—	Disable
			0001	I/O-Pad	J[4]
			0010	I/O-Pad	J[9]
			0011-1111	—	Reserved
DSPI_0	SIN	IMCR[41]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[7]
			0010-1111	—	Reserved
DSPI_1	SIN	IMCR[44]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[8]
			0010-1111	—	Reserved
DSPI_2	SIN	IMCR[47]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[13]
			0010	I/O-Pad	A[2]
			0011-1111	—	Reserved
DSPI_2	SCK	IMCR[48]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[0]
			0010	I/O-Pad	A[11]
			0011-1111	—	Reserved
DSPI_2	SC0	IMCR[49]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[3]
			0010	I/O-Pad	A[10]
			0011-1111	—	Reserved
DSPI_3	SIN	IMCR[50]	0000 (Default)	—	Disable
			0001	I/O-Pad	J[1]
			0010	I/O-Pad	D[7]
			0011	I/O-Pad	D[14]
			0100	I/O-Pad	E[15]
			0101-1111	—	Reserved
DSPI_3	SCK	IMCR[51]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[6]
			0010	I/O-Pad	D[11]
			0011	I/O-Pad	E[13]
			0100	I/O-Pad	I[15]
			0101-1111	—	Reserved
DSPI_3	CS0	IMCR[52]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[11]
			0010	I/O-Pad	D[10]
			0011	I/O-Pad	F[5]

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0100	I/O-Pad	I[14]
			0101-1111	—	Reserved
eTimer_0	ETC0	IMCR[59]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[10]
			0010	I/O-Pad	A[0]
			0011-1111	—	Reserved
eTimer_0	ETC1	IMCR[60]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[11]
			0010	I/O-Pad	A[1]
			0011-1111	—	Reserved
eTimer_0	ETC2	IMCR[61]	0000 (Default)	—	Disable
			0001	I/O-Pad	F[0]
			0010	I/O-Pad	A[2]
			0011-1111	—	Reserved
eTimer_0	ETC3	IMCR[62]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[14]
			0010	I/O-Pad	A[3]
			0011-1111	—	Reserved
eTimer_0	ETC4	IMCR[63]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[14]
			0010	I/O-Pad	G[3]
			0011	I/O-Pad	A[4]
			0100	I/O-Pad	C[11]
			0101-1111	—	Reserved
eTimer_0	ETC5	IMCR[64]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[8]
			0010	I/O-Pad	G[4]
			0011	I/O-Pad	C[12]
			0100	I/O-Pad	E[13]
			0101-1111	—	Reserved
eTimer_1	ETC0	IMCR[65]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[4]
			0010	I/O-Pad	C[15]
			0011-1111	—	Reserved
eTimer_1	ETC1	IMCR[66]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[13]
			0010	I/O-Pad	D[0]
			0011-1111	—	Reserved
eTimer_1	ETC2	IMCR[67]	0000 (Default)	—	Disable

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0001	I/O-Pad	B[0]
			0010	I/O-Pad	C[14]
			0011	I/O-Pad	D[1]
			0100-1111	—	Reserved
eTimer_1	ETC3	IMCR[68]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[1]
			0010	I/O-Pad	D[2]
			0011	I/O-Pad	F[12]
			0100-1111	—	Reserved
eTimer_1	ETC4	IMCR[69]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[14]
			0010	I/O-Pad	D[3]
			0011	I/O-Pad	D[8]
			0100	I/O-Pad	F[13]
			0101-1111	—	Reserved
eTimer_1	ETC5	IMCR[70]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[5]
			0010	I/O-Pad	A[15]
			0011	I/O-Pad	D[4]
			0100	I/O-Pad	E[14]
			0101-1111	—	Reserved
eTimer_2	ETC0	IMCR[71]	0000 (Default)	—	Disable
			0001	I/O-Pad	H[4]
			0010	I/O-Pad	I[0]
			0011-1111	—	Reserved
eTimer_2	ETC1	IMCR[72]	0000 (Default)	—	Disable
			0001	I/O-Pad	H[7]
			0010	I/O-Pad	I[1]
			0011-1111	—	Reserved
eTimer_2	ETC2	IMCR[73]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[6]
			0010	I/O-Pad	H[10]
			0011	I/O-Pad	I[2]
			0100	I/O-Pad	J[8]
			0101-1111	—	Reserved
eTimer_2	ETC3	IMCR[74]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[7]
			0010	I/O-Pad	H[13]
			0011	I/O-Pad	I[3]

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0100-1111	—	Reserved
eTimer_2	ETC4	IMCR[75]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[8]
			0010	I/O-Pad	H[14]
			0011	I/O-Pad	I[9]
			0100	I/O-Pad	J[8]
			0101-1111	—	Reserved
eTimer_2	ETC5	IMCR[76]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[9]
			0010	I/O-Pad	H[15]
			0011	I/O-Pad	I[10]
			0100	I/O-Pad	J[9]
			0101-1111	—	Reserved
FlexPWM_0	FAULT0	IMCR[83]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[9]
			0010	I/O-Pad	A[13]
			0011	I/O-Pad	G[8]
			0100-1111	—	Reserved
FlexPWM_0	FAULT1	IMCR[84]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[10]
			0010	I/O-Pad	D[6]
			0011	I/O-Pad	G[9]
			0100-1111	—	Reserved
FlexPWM_0	FAULT2	IMCR[85]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[5]
			0010	I/O-Pad	G[10]
			0011-1111	—	Reserved
FlexPWM_0	FAULT3	IMCR[86]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[5]
			0010	I/O-Pad	D[8]
			0011	I/O-Pad	G[11]
			0100-1111	—	Reserved
FlexPWM_0	EXT_SYNC	IMCR[87]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[13]
			0010	I/O-Pad	C[15]
			0011-1111	—	Reserved
FlexPWM_0	A0	IMCR[88]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[11]
			0010	I/O-Pad	D[10]

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0011-1111	—	Reserved
FlexPWM_0	B0	IMCR[89]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[10]
			0010	I/O-Pad	D[11]
			0011-1111	—	Reserved
FlexPWM_0	A1	IMCR[91]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[7]
			0010	I/O-Pad	C[15]
			0011	I/O-Pad	F[0]
			0100-1111	—	Reserved
FlexPWM_0	B1	IMCR[92]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[6]
			0010	I/O-Pad	D[0]
			0011	I/O-Pad	D[14]
			0100-1111	—	Reserved
FlexPWM_0	X1	IMCR[93]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[4]
			0010	I/O-Pad	D[12]
			0011-1111	—	Reserved
FlexPWM_0	A2	IMCR[94]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[11]
			0010	I/O-Pad	A[12]
			0011	I/O-Pad	G[3]
			0100-1111	—	Reserved
FlexPWM_0	B2	IMCR[95]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[12]
			0010	I/O-Pad	A[13]
			0011	I/O-Pad	G[4]
			0100-1111	—	Reserved
FlexPWM_0	X2	IMCR[96]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[10]
			0010	I/O-Pad	G[2]
			0011-1111	—	Reserved
FlexPWM_0	A3	IMCR[97]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[2]
			0010	I/O-Pad	C[10]
			0011	I/O-Pad	D[3]
			0100	I/O-Pad	G[6]
			0101-1111	—	Reserved

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
FlexPWM_0	B3	IMCR[98]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[3]
			0010	I/O-Pad	A[9]
			0011	I/O-Pad	D[4]
			0100	I/O-Pad	G[7]
			0101-1111	—	Reserved
FlexPWM_0	X3	IMCR[99]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[2]
			0010	I/O-Pad	D[6]
			0011	I/O-Pad	G[5]
			0100-1111	—	Reserved
FlexPWM_1	FAULT0	IMCR[100]	0000 (Default)	—	Disable
			0001	I/O-Pad	I[0]
			0010-1111	—	Reserved
FlexPWM_1	FAULT1	IMCR[101]	0000 (Default)	—	Disable
			0001	I/O-Pad	I[1]
			0010-1111	—	Reserved
FlexPWM_1	FAULT2	IMCR[102]	0000 (Default)	—	Disable
			0001	I/O-Pad	I[2]
			0010-1111	—	Reserved
FlexPWM_1	FAULT3	IMCR[103]	0000 (Default)	—	Disable
			0001	I/O-Pad	I[3]
			0010-1111	—	Reserved
FlexPWM_1	A0	IMCR[105]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[13]
			0010	I/O-Pad	H[5]
			0011-1111	—	Reserved
FlexPWM_1	B0	IMCR[106]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[14]
			0010	I/O-Pad	H[6]
			0011-1111	—	Reserved
FlexPWM_1	A1	IMCR[109]	0000 (Default)	—	Disable
			0001	I/O-Pad	F[12]
			0010	I/O-Pad	H[8]
			0011-1111	—	Reserved
FlexPWM_1	B1	IMCR[110]	0000 (Default)	—	Disable
			0001	I/O-Pad	F[13]
			0010	I/O-Pad	H[9]
			0011-1111	—	Reserved

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
FlexPWM_1	A2	IMCR[112]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[4]
			0010	I/O-Pad	H[11]
			0011-1111	—	Reserved
FlexPWM_1	B2	IMCR[113]	0000	—	Disable
			0001	I/O-Pad	E[14]
			0010	I/O-Pad	H[12]
			0011-1111	—	Reserved
FlexRay	FR_A_RX	IMCR[136]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[1]
			0010-1111	—	Reserved
FlexRay	FR_B_RX	IMCR[137]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[2]
			0010-1111	—	Reserved
LIN_0	RXD	IMCR[165]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[3]
			0010	I/O-Pad	B[7]
			0011-1111	—	Reserved
LIN_1	RXD	IMCR[166]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[13]
			0010	I/O-Pad	D[12]
			0011	I/O-Pad	F[15]
			0100-1111	—	Reserved
MC_RGM	ABS0	IMCR[169]	0000 (Default)	I/O-Pad	A[2]
			0001	—	Disable
			0010-1111	—	Reserved
MC_RGM	ABS2	IMCR[171]	0000 (Default)	I/O-Pad	A[3]
			0001	—	Disable
			0010-1111	—	Reserved
MC_RGM	FAB	IMCR[172]	0000 (Default)	I/O-Pad	A[4]
			0001	—	Disable
			0010-1111	—	Reserved
SIUL	REQ0	IMCR[173]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[0]
			0010-1111	—	Reserved
SIUL	REQ1	IMCR[174]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[1]
			0010-1111	—	Reserved
SIUL	REQ2	IMCR[175]	0000 (Default)	—	Disable

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0001	I/O-Pad	A[2]
			0010-1111	—	Reserved
SIUL	REQ3	IMCR[176]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[3]
			0010-1111	—	Reserved
SIUL	REQ4	IMCR[177]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[4]
			0010-1111	—	Reserved
SIUL	REQ5	IMCR[178]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[5]
			0010-1111	—	Reserved
SIUL	REQ6	IMCR[179]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[6]
			0010-1111	—	Reserved
SIUL	REQ7	IMCR[180]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[7]
			0010-1111	—	Reserved
SIUL	REQ8	IMCR[181]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[8]
			0010-1111	—	Reserved
SIUL	REQ9	IMCR[182]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[10]
			0010-1111	—	Reserved
SIUL	REQ10	IMCR[183]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[11]
			0010-1111	—	Reserved
SIUL	REQ11	IMCR[184]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[12]
			0010-1111	—	Reserved
SIUL	REQ12	IMCR[185]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[13]
			0010-1111	—	Reserved
SIUL	REQ13	IMCR[186]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[14]
			0010-1111	—	Reserved
SIUL	REQ14	IMCR[187]	0000 (Default)	—	Disable
			0001	I/O-Pad	A[15]
			0010-1111	—	Reserved
SIUL	REQ15	IMCR[188]	0000 (Default)	—	Disable

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0001	I/O-Pad	B[0]
			0010-1111	—	Reserved
SIUL	REQ16	IMCR[189]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[1]
			0010-1111	—	Reserved
SIUL	REQ17	IMCR[190]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[2]
			0010-1111	—	Reserved
SIUL	REQ18	IMCR[191]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[6]
			0010-1111	—	Reserved
SIUL	REQ19	IMCR[192]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[14]
			0010-1111	—	Reserved
SIUL	REQ20	IMCR[193]	0000 (Default)	—	Disable
			0001	I/O-Pad	B[15]
			0010-1111	—	Reserved
SIUL	REQ21	IMCR[194]	0000 (Default)	—	Disable
			0001	I/O-Pad	G[8]
			0010-1111	—	Reserved
SIUL	REQ22	IMCR[195]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[4]
			0010-1111	—	Reserved
SIUL	REQ23	IMCR[196]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[5]
			0010-1111	—	Reserved
SIUL	REQ24	IMCR[197]	0000 (Default)	—	Disable
			0001	I/O-Pad	C[6]
			0010-1111	—	Reserved
SIUL	REQ25	IMCR[198]	0000 (Default)	—	Disable
			0001	I/O-Pad	E[13]
			0010-1111	—	Reserved
SIUL	REQ26	IMCR[199]	0000 (Default)	—	Disable
			0001	I/O-Pad	E[14]
			0010-1111	—	Reserved
SIUL	REQ27	IMCR[200]	0000 (Default)	—	Disable
			0001	I/O-Pad	E[15]
			0010-1111	—	Reserved
SIUL	REQ28	IMCR[201]	0000 (Default)	—	Disable

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0001	I/O-Pad	F[0]
			0010-1111	—	Reserved
SIUL	REQ29	IMCR[202]	0000 (Default)	—	Disable
			0001	I/O-Pad	G[9]
			0010-1111	—	Reserved
SIUL	REQ30	IMCR[203]	0000 (Default)	—	Disable
			0001	I/O-Pad	F[12]
			0010-1111	—	Reserved
SIUL	REQ31	IMCR[204]	0000 (Default)	—	Disable
			0001	I/O-Pad	F[13]
			0010-1111	—	Reserved
SENT_0	SENT_RX[0]	IMCR[205]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[5]
			0010	I/O-Pad	I[7]
			0011	I/O-Pad	G[8]
			0100-1111	—	Reserved
SENT_0	SENT_RX[1]	IMCR[206]	0000 (Default)	—	Disable
			0001	I/O-Pad	I[11]
			0010	I/O-Pad	J[5]
			0011	I/O-Pad	A[9]
			0100	I/O-Pad	G[10]
			0101-1111	—	Reserved
SENT_1	SENT_RX[0]	IMCR[213]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[7]
			0010	I/O-Pad	I[8]
			0011	I/O-Pad	G[9]
			0100	I/O-Pad	C[12]
			0101-1111	—	Reserved
SENT_1	SENT_RX[1]	IMCR[214]	0000 (Default)	—	Disable
			0001	I/O-Pad	I[12]
			0010	I/O-Pad	J[6]
			0011	I/O-Pad	A[10]
			0100	I/O-Pad	G[11]
			0101-1111	—	Reserved
ENET_0	RX_CLK	IMCR[224]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[8]
			0010-1111	—	Reserved
ENET_0	RX_DV	IMCR[225]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[7]

Table continues on the next page...

Table 4-9. Peripheral muxing (continued)

Destination peripheral	Destination functions	IMCR number	IMCR[SSS] field value	Source peripherals	Source functions
			0010-1111	—	Reserved
ENET_0	RX_D0	IMCR[226]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[6]
			0010-1111	—	Reserved
ENET_0	RX_D1	IMCR[227]	0000 (Default)	—	Disable
			0001	I/O-Pad	D[5]
			0010-1111	—	Reserved
ENET_0	RX_D2	IMCR[228]	0000 (Default)	—	Disable
			0001	I/O-Pad	H[8]
			0010-1111	—	Reserved
ENET_0	RX_D3	IMCR[229]	0000 (Default)	—	Disable
			0001	I/O-Pad	J[9]
			0010-1111	—	Reserved
ENET_0	COL	IMCR[230]	0000 (Default)	—	Disable
			0001	I/O-Pad	H[5]
			0010-1111	—	Reserved
ENET_0	CRS	IMCR[231]	0000 (Default)	—	Disable
			0001	I/O-Pad	H[4]
			0010-1111	—	Reserved
ENET_0	RX_ER	IMCR[232]	0000 (Default)	—	Disable
			0001	I/O-Pad	I[1]
			0010-1111	—	Reserved
ENET_0	TX_CLK	IMCR[233]	0000 (Default)	—	Disable
			0001	I/O-Pad	G[8]
			0010-1111	—	Reserved

1. (Default) = configuration after reset
2. Selecting an alternate function with a 'Reserved' source function causes the pin to enter a null state (Input buffer and Output buffer enables both at 0).

Table 4-10. Peripheral muxing example

SSS field value in IMCR[214]	Result
0001	I/O-Pad I[12] is connected to SENT_1 Receive input SENT_RX[1]
0010	I/O-Pad J[6] is connected to SENT_1 Receive input SENT_RX[1]

See Table 4-8 concerning the availability of port pins on the packages.

4.3.7 Pins with analog functions

The following table describes pins with analog functions.

Table 4-11. Pins with analog functions

Pin	144LQFP	257MAPBGA	Connected Analog IPs	Analog Function[1]	MSCR#
B[7]	43	R5	ADC0	AN[0]	23
B[8]	47	P7	ADC0	AN[1]	24
B[9]	52	U7	ADC0/ADC1	AN[11]	25
B[10]	53	R8	ADC0/ADC1	AN[12]	26
B[11]	54	T8	ADC0/ADC1	AN[13]	27
B[12]	55	U8	ADC0/ADC1	AN[14]	28
B[13]	60	R10	ADC1	AN[0]	29
B[14]	64	P11	ADC1	AN[1]	30
B[15]	62	R11	ADC1	AN[2]	31
C[0]	66	R12	ADC1	AN[3]	32
C[1]	41	T4	ADC0	AN[2]	33
C[2]	45	U5	ADC0	AN[3]	34
D[7]	37	R4	Sine Wave Generator Out	anaout	NA[2]
E[0]	68	T13	ADC1	AN[5]	64
			ADC3	AN[4]	
E[2]	49	U6	ADC0	AN[5]	66
E[4]	42	U4	ADC0	AN[7]	68
E[5]	44	T5	ADC0	AN[8]	69
E[6]	46	R6	ADC0/ADC2	AN[4]	70
E[7]	48	T6	ADC0	AN[6]	71
E[9]	61	U10	ADC1	AN[7]	73
			ADC3	AN[6]	
E[10]	63	T11	ADC1	AN[8]	74
			ADC3	AN[7]	
E[11]	65	U11	ADC1	AN[4]	75
			ADC3	AN[3]	
E[12]	67	T12	ADC1	AN[6]	76
			ADC3	AN[5]	
J[5]		P8	ADC2/ADC3	AN[0]	149
J[6]		P9	ADC2/ADC3	AN[1]	150
J[7]		P10	ADC2/ADC3	AN[2]	151

[1] For pin multiplexed ADC functions - both ADCs will have the same channel number unless otherwise specified.

[2] The Sine Wave Generator must be enabled to switch on the analog output function.

4.3.8 Mapping of ports to PGPDO registers

The following table shows the mapping of the GPIO ports to the Parallel GPIO Pad Data Out (PGPDO) registers. See the dedicated SIUL2 chapter for more information.

Table 4-12. Mapping of ports to PGPDO registers

Port[1]	Port width	PGPDO field[2]	Address
A	16	PPDO[0]	0xFFFC1700
B	16	PPDO[1]	0xFFFC1702
C	16	PPDO[2]	0xFFFC1704
D	16	PPDO[3]	0xFFFC1706
E	16	PPDO[4]	0xFFFC1708
F	16	PPDO[5]	0xFFFC170A
G	16	PPDO[6]	0xFFFC170C
H	16	PPDO[7]	0xFFFC170E
I	16	PPDO[8]	0xFFFC1710
J	16	PPDO[9]	0xFFFC1712

[1] See Ports - Not Implemented table for unimplemented port pins.
[2] All fields are 16 bits wide.

The following table shows the mapping of the GPIO ports to Parallel GPIO Pad Data In (PGPDI) registers. See the dedicated SIUL2 chapter for more information

Table 4-13. Mapping of ports to PGPDI registers

Port[1]	Port width	PGPDI field[2]	Address
A	16	PPDI[0]	0xFFFC1740
B	16	PPDI[1]	0xFFFC1742
C	16	PPDI[2]	0xFFFC1744
D	16	PPDI[3]	0xFFFC1746
E	16	PPDI[4]	0xFFFC1748
F	16	PPDI[5]	0xFFFC174A
G	16	PPDI[6]	0xFFFC174C
H	16	PPDI[7]	0xFFFC174E
I	16	PPDI[8]	0xFFFC1750
J	16	PPDI[9]	0xFFFC1752

[1] See Ports - Not Implemented table for unimplemented port pins.
[2] All fields are 16 bits wide.

4.3.9 Address map and configuration for MSCR/IMCR registers

The following tables provide the memory map for the MSCRs and IMCRs used to configure alternative modes and peripheral input muxing. See the SIUL2 chapter for the registers' layouts.

Table 4-16 shows the default settings for the pad control fields and which pad control fields are programmable (see Table 4-15 for conventions).

The default settings (configuration after reset) for the MSCR SSS fields are documented in Table 4-7.

Table 4-17 shows the IMCRs by peripheral: destination function, IMCR number, IMCR address, default configuration and programmability of the INV field, and default value for the SSS field.

Table 4-14. MSCR pad control fields

MSCR field	Brief description
SRC[1:0]	Slew Rate Control
OBE	Output Buffer Enable
ODE	Open Drain Enable
SMC	Safe Mode Control
APC	Analog Pad Control
IBE	Input Buffer Enable
HYS	Input Hysteresis
PUS	Pull Select: 0 = Pulldown, 1 = Pullup
PUE	Pull Enable: 0 = Disabled, 1 = Enabled
INV	Invert

Table 4-15 describes conventions used in Table 4-16 and Table 4-17.

Table 4-15. Conventions used in following MSCR and IMCR tables

Abbreviation	Configuration during/after reset	Programmable by user?
D	Disabled	No: permanently disabled or not implemented
E	Enabled	No: permanently enabled
P(0)	Reset value is 0	Yes
P(1)	Reset value is 1	Yes
P(D)	Disabled	Yes
P(E)	Enabled	Yes
P(PD)	Pulldown	Yes
P(PU)	Pullup	Yes
PD	Pulldown	No: permanently selected
PU	Pullup	No: permanently selected

Table 4-16. MSCR addresses and default settings

Data Pin	MSCR #	Hex address	SRC[1]	SRC[0][1]	OBE	ODE	SMC	APC	IBE	HYS	PUS	PUE	INV
A[0]	0	FFFC_0240	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[1]	1	FFFC_0244	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[2]	2	FFFC_0248	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(E)	P(E)	P(PD)	P(E)	P(D)
A[3]	3	FFFC_024c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(E)	P(E)	P(PD)	P(E)	P(D)
A[4]	4	FFFC_0250	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(E)	P(E)	P(PD)	P(E)	P(D)
A[5]	5	FFFC_0254	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[6]	6	FFFC_0258	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[7]	7	FFFC_025c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[8]	8	FFFC_0260	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[9]	9	FFFC_0264	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[10]	10	FFFC_0268	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[11]	11	FFFC_026c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[12]	12	FFFC_0270	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[13]	13	FFFC_0274	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[14]	14	FFFC_0278	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
A[15]	15	FFFC_027c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
B[0]	16	FFFC_0280	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
B[1]	17	FFFC_0284	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
B[2]	18	FFFC_0288	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
B[3]	19	FFFC_028c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
B[4]	20	FFFC_0290	P(1)	P(1)	P(E)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
B[5]	21	FFFC_0294	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(E)	P(E)	P(PU)	P(E)	P(D)
B[6]	22	FFFC_0298	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
B[7]	23	FFFC_029c	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
B[8]	24	FFFC_02a0	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
B[9]	25	FFFC_02a4	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
B[10]	26	FFFC_02a8	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
B[11]	27	FFFC_02ac	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
B[12]	28	FFFC_02b0	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
B[13]	29	FFFC_02b4	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
B[14]	30	FFFC_02b8	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
B[15]	31	FFFC_02bc	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
C[0]	32	FFFC_02c0	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
C[1]	33	FFFC_02c4	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
C[2]	34	FFFC_02c8	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
C[4]	36	FFFC_02d0	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
C[5]	37	FFFC_02d4	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
C[6]	38	FFFC_02d8	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)

Table continues on the next page...

Table 4-16. MSCR addresses and default settings (continued)

Data Pin	MSCR #	Hex address	SRC[1]	SRC[0][1]	OBE	ODE	SMC	APC	IBE	HYS	PUS	PUE	INV
C[7]	39	FFFC_02dc	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
C[10]	42	FFFC_02e8	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
C[11]	43	FFFC_02ec	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
C[12]	44	FFFC_02f0	P(0)	P(0)	P(D)	P(D)	D	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
C[13]	45	FFFC_02f4	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
C[14]	46	FFFC_02f8	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
C[15]	47	FFFC_02fc	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[0]	48	FFFC_0300	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[1]	49	FFFC_0304	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[2]	50	FFFC_0308	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[3]	51	FFFC_030c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[4]	52	FFFC_0310	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[5]	53	FFFC_0314	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[6]	54	FFFC_0318	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[7]	55	FFFC_031c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[8]	56	FFFC_0320	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[9]	57	FFFC_0324	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[10]	58	FFFC_0328	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[11]	59	FFFC_032c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[12]	60	FFFC_0330	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
D[14]	62	FFFC_0338	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
E[0]	64	FFFC_0340	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[2]	66	FFFC_0348	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[4]	68	FFFC_0350	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[5]	69	FFFC_0354	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[6]	70	FFFC_0358	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[7]	71	FFFC_035c	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[9]	73	FFFC_0364	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[10]	74	FFFC_0368	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[11]	75	FFFC_036c	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[12]	76	FFFC_0370	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
E[13]	77	FFFC_0374	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
E[14]	78	FFFC_0378	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
E[15]	79	FFFC_037c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[0]	80	FFFC_0380	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[3]	83	FFFC_038c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[4]	84	FFFC_0390	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[5]	85	FFFC_0394	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[6]	86	FFFC_0398	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)

Table continues on the next page...

Table 4-16. MSCR addresses and default settings (continued)

Data Pin	MSCR #	Hex address	SRC[1]	SRC[0][1]	OBE	ODE	SMC	APC	IBE	HYS	PUS	PUE	INV
F[7]	87	FFFC_039c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[8]	88	FFFC_03a0	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[9]	89	FFFC_03a4	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[10]	90	FFFC_03a8	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[11]	91	FFFC_03ac	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[12]	92	FFFC_03b0	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[13]	93	FFFC_03b4	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[14]	94	FFFC_03b8	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
F[15]	95	FFFC_03bc	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[2]	98	FFFC_03c8	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[3]	99	FFFC_03cc	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[4]	100	FFFC_03d0	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[5]	101	FFFC_03d4	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[6]	102	FFFC_03d8	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[7]	103	FFFC_03dc	P(0)	P(0)	P(D)	P(D)	D	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[8]	104	FFFC_03e0	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[9]	105	FFFC_03e4	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[10]	106	FFFC_03e8	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
G[11]	107	FFFC_03ec	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[4]	116	FFFC_0410	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[5]	117	FFFC_0414	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[6]	118	FFFC_0418	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[7]	119	FFFC_041c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[8]	120	FFFC_0420	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[9]	121	FFFC_0424	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[10]	122	FFFC_0428	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[11]	123	FFFC_042c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[12]	124	FFFC_0430	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[13]	125	FFFC_0434	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[14]	126	FFFC_0438	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
H[15]	127	FFFC_043c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[0]	128	FFFC_0440	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[1]	129	FFFC_0444	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[2]	130	FFFC_0448	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[3]	131	FFFC_044c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
RDY_B/ I[4]	132	FFFC_0450	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[5]	133	FFFC_0454	P(0)	P(0)	P(D)	P(D)	D	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[6]	134	FFFC_0458	P(0)	P(0)	P(D)	P(D)	D	NA	P(D)	P(E)	P(PD)	P(D)	P(D)

Table continues on the next page...

Table 4-16. MSCR addresses and default settings (continued)

Data Pin	MSCR #	Hex address	SRC[1]	SRC[0][1]	OBE	ODE	SMC	APC	IBE	HYS	PUS	PUE	INV
I[7]	135	FFFC_045c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[8]	136	FFFC_0460	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[9]	137	FFFC_0464	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[10]	138	FFFC_0468	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[11]	139	FFFC_046c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[12]	140	FFFC_0470	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[13]	141	FFFC_0474	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[14]	142	FFFC_0478	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
I[15]	143	FFFC_047c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
J[0]	144	FFFC_0480	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
J[1]	145	FFFC_0484	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
J[2]	146	FFFC_0488	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
J[3]	147	FFFC_048c	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
J[4]	148	FFFC_0490	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
J[5]	149	FFFC_0494	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
J[6]	150	FFFC_0498	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
J[7]	151	FFFC_049c	NA	NA	NA	NA	D	P(D)	P(D)	NA	NA	D	D
J[8]	152	FFFC_04a0	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
J[9]	153	FFFC_04a4	P(0)	P(0)	P(D)	P(D)	P(E)	NA	P(D)	P(E)	P(PD)	P(D)	P(D)
NMI_B	154	FFFC_04a8	0	0	D	D	D	NA	E	P(E)	PU	E	D

[1] Slew Rate Control bits controlling both the slew rate of the pad and the drive strength. See the SIUL chapter for a complete bit description.

Table 4-17. IMCR addresses and default settings

Destination peripheral	Destination function	IMCR number	Hex address	INV configuration[1]	SSS default value
CAN_0	RXD	32	FFFC_0AC0	P(D)	0000
CAN_1	RXD	33	FFFC_0AC4	P(D)	0000
CAN_2	RXD	34	FFFC_0AC8	P(D)	0000
CTU_0	EXT_IN	38	FFFC_0AD8	P(D)	0000
CTU_1	EXT_IN	39	FFFC_0ADC	P(D)	0000
DSPI_0	SIN	41	FFFC_0AE4	P(D)	0000
DSPI_1	SIN	44	FFFC_0AF0	P(D)	0000
DSPI_2	SIN	47	FFFC_0AFC	P(D)	0000
DSPI_2	SCK	48	FFFC_0B00	P(D)	0000
DSPI_2	CS0	49	FFFC_0B04	P(D)	0000
DSPI_3	SIN	50	FFFC_0B08	P(D)	0000
DSPI_3	SCK	51	FFFC_0B0C	P(D)	0000

Table continues on the next page...

Table 4-17. IMCR addresses and default settings (continued)

Destination peripheral	Destination function	IMCR number	Hex address	INV configuration[1]	SSS default value
DSPI_3	CS0	52	FFFC_0B10	P(D)	0000
eTimer_0	ETC0	59	FFFC_0B2C	P(D)	0000
eTimer_0	ETC1	60	FFFC_0B30	P(D)	0000
eTimer_0	ETC2	61	FFFC_0B34	P(D)	0000
eTimer_0	ETC3	62	FFFC_0B38	P(D)	0000
eTimer_0	ETC4	63	FFFC_0B3C	P(D)	0000
eTimer_0	ETC5	64	FFFC_0B40	P(D)	0000
eTimer_1	ETC0	65	FFFC_0B44	P(D)	0000
eTimer_1	ETC1	66	FFFC_0B48	P(D)	0000
eTimer_1	ETC2	67	FFFC_0B4C	P(D)	0000
eTimer_1	ETC3	68	FFFC_0B50	P(D)	0000
eTimer_1	ETC4	69	FFFC_0B54	P(D)	0000
eTimer_1	ETC5	70	FFFC_0B58	P(D)	0000
eTimer_2	ETC0	71	FFFC_0B5C	P(D)	0000
eTimer_2	ETC1	72	FFFC_0B60	P(D)	0000
eTimer_2	ETC2	73	FFFC_0B64	P(D)	0000
eTimer_2	ETC3	74	FFFC_0B68	P(D)	0000
eTimer_2	ETC4	75	FFFC_0B6C	P(D)	0000
eTimer_2	ETC5	76	FFFC_0B70	P(D)	0000
FlexPWM_0	FAULT0	83	FFFC_0B8C	P(D)	0000
FlexPWM_0	FAULT1	84	FFFC_0B90	P(D)	0000
FlexPWM_0	FAULT2	85	FFFC_0B94	P(D)	0000
FlexPWM_0	FAULT3	86	FFFC_0B98	P(D)	0000
FlexPWM_0	EXT_SYNC	87	FFFC_0B9C	P(D)	0000
FlexPWM_0	A0	88	FFFC_0BA0	P(D)	0000
FlexPWM_0	B0	89	FFFC_0BA4	P(D)	0000
FlexPWM_0	A1	91	FFFC_0BAC	P(D)	0000
FlexPWM_0	B1	92	FFFC_0BB0	P(D)	0000
FlexPWM_0	X1	93	FFFC_0BB4	P(D)	0000
FlexPWM_0	A2	94	FFFC_0BB8	P(D)	0000
FlexPWM_0	B2	95	FFFC_0BBC	P(D)	0000
FlexPWM_0	X2	96	FFFC_0BC0	P(D)	0000
FlexPWM_0	A3	97	FFFC_0BC4	P(D)	0000
FlexPWM_0	B3	98	FFFC_0BC8	P(D)	0000
FlexPWM_0	X3	99	FFFC_0BCC	P(D)	0000
FlexPWM_1	FAULT0	100	FFFC_0BD0	P(D)	0000
FlexPWM_1	FAULT1	101	FFFC_0BD4	P(D)	0000
FlexPWM_1	FAULT2	102	FFFC_0BD8	P(D)	0000
FlexPWM_1	FAULT3	103	FFFC_0BDC	P(D)	0000

Table continues on the next page...

Table 4-17. IMCR addresses and default settings (continued)

Destination peripheral	Destination function	IMCR number	Hex address	INV configuration[1]	SSS default value
FlexPWM_1	A0	105	FFFC_0BE4	P(D)	0000
FlexPWM_1	B0	106	FFFC_0BE8	P(D)	0000
FlexPWM_1	A1	109	FFFC_0BF4	P(D)	0000
FlexPWM_1	B1	110	FFFC_0BF8	P(D)	0000
FlexPWM_1	A2	112	FFFC_0C00	P(D)	0000
FlexPWM_1	B2	113	FFFC_0C04	P(D)	0000
FlexRay	FR_A_RX	136	FFFC_0C60	P(D)	0000
FlexRay	FR_B_RX	137	FFFC_0C64	P(D)	0000
LIN_0	RXD	165	FFFC_0CD4	P(D)	0000
LIN_1	RXD	166	FFFC_0CD8	P(D)	0000
MC_RGM	ABS0	169	FFFC_0CE4	P(D)	0000
MC_RGM	ABS2	171	FFFC_0CEC	P(D)	0000
MC_RGM	FAB	172	FFFC_0CF0	P(D)	0000
SIUL2	REQ0	173	FFFC_0CF4	P(D)	0000
SIUL2	REQ1	174	FFFC_0CF8	P(D)	0000
SIUL2	REQ2	175	FFFC_0CFC	P(D)	0000
SIUL2	REQ3	176	FFFC_0D00	P(D)	0000
SIUL2	REQ4	177	FFFC_0D04	P(D)	0000
SIUL2	REQ5	178	FFFC_0D08	P(D)	0000
SIUL2	REQ6	179	FFFC_0D0C	P(D)	0000
SIUL2	REQ7	180	FFFC_0D10	P(D)	0000
SIUL2	REQ8	181	FFFC_0D14	P(D)	0000
SIUL2	REQ9	182	FFFC_0D18	P(D)	0000
SIUL2	REQ10	183	FFFC_0D1C	P(D)	0000
SIUL2	REQ11	184	FFFC_0D20	P(D)	0000
SIUL2	REQ12	185	FFFC_0D24	P(D)	0000
SIUL2	REQ13	186	FFFC_0D28	P(D)	0000
SIUL2	REQ14	187	FFFC_0D2C	P(D)	0000
SIUL2	REQ15	188	FFFC_0D30	P(D)	0000
SIUL2	REQ16	189	FFFC_0D34	P(D)	0000
SIUL2	REQ17	190	FFFC_0D38	P(D)	0000
SIUL2	REQ18	191	FFFC_0D3C	P(D)	0000
SIUL2	REQ19	192	FFFC_0D40	P(D)	0000
SIUL2	REQ20	193	FFFC_0D44	P(D)	0000
SIUL2	REQ21	194	FFFC_0D48	P(D)	0000
SIUL2	REQ22	195	FFFC_0D4C	P(D)	0000
SIUL2	REQ23	196	FFFC_0D50	P(D)	0000
SIUL2	REQ24	197	FFFC_0D54	P(D)	0000
SIUL2	REQ25	198	FFFC_0D58	P(D)	0000

Table continues on the next page...

Table 4-17. IMCR addresses and default settings (continued)

Destination peripheral	Destination function	IMCR number	Hex address	INV configuration[1]	SSS default value
SIUL2	REQ26	199	FFFC_0D5C	P(D)	0000
SIUL2	REQ27	200	FFFC_0D60	P(D)	0000
SIUL2	REQ28	201	FFFC_0D64	P(D)	0000
SIUL2	REQ29	202	FFFC_0D68	P(D)	0000
SIUL2	REQ30	203	FFFC_0D6C	P(D)	0000
SIUL2	REQ31	204	FFFC_0D70	P(D)	0000
SENT_0	SENT_RX[0]	205	FFFC_0D74	P(D)	0000
SENT_0	SENT_RX[1]	206	FFFC_0D78	P(D)	0000
SENT_1	SENT_RX[0]	213	FFFC_0D94	P(D)	0000
SENT_1	SENT_RX[1]	214	FFFC_0D98	P(D)	0000
ENET_0	RX_CLK	224	FFFC_0DC0	P(D)	0000
ENET_0	RX_DV	225	FFFC_0DC4	P(D)	0000
ENET_0	RX_D0	226	FFFC_0DC8	P(D)	0000
ENET_0	RX_D1	227	FFFC_0DCC	P(D)	0000
ENET_0	RX_D2	228	FFFC_0DD0	P(D)	0000
ENET_0	RX_D3	229	FFFC_0DD4	P(D)	0000
ENET_0	COL	230	FFFC_0DD8	P(D)	0000
ENET_0	CRS	231	FFFC_0DDC	P(D)	0000
ENET_0	RX_ER	232	FFFC_0DE0	P(D)	0000
ENET_0	TX_CLK	233	FFFC_0DE4	P(D)	0000

[1] P(D) means the Invert function is disabled by default but can be programmed by the user.

Chapter 5
Memory Map

5.1 Memory Map

The following table shows the device memory map for the MPC5744P.

All addresses on the MPC5744P, including many that are reserved, are identified in the table. The addresses represent the physical addresses assigned to each region or module name. All memory not listed in this table is reserved, and access to those locations may produce undesirable results.

Table 5-1. Overview memory map

Start address	End address	Allocated size	Used size	PCTL number	Description
Flash memory—see Table 5-4 for details					
0x00000000	0x003FFFFF	-			Reserved
0x00400000	0x00403FFF	16 KB		-	UTest NVM block - no overlay See Table 5-6 for details
0x00404000	0x007FFFFF	-			Reserved
0x00800000	0x00817FFF	96 KB		-	Data flash memory blocks - no overlay
0x00818000	0x009FFFFF	-			Reserved
0x00A00000	0x00FFFFFF	6 MB	416 KB	-	Small and medium flash memory blocks - no overlay
0x01000000	0x01FFFFFF	16 MB	2048 KB	-	Large flash memory blocks - no overlay
0x02000000	0x089FFFFF	106 MB			Reserved
0x08A00000	0x08FFFFFF	6 MB	416 KB	-	Mirror small and medium flash memory blocks - no overlay

Table continues on the next page...

Table 5-1. Overview memory map (continued)

Start address	End address	Allocated size	Used size	PCTL number	Description
0x09000000	0x09FFFFFF	16 MB	2048 KB	-	Mirror large flash memory blocks - no overlay
0x0A000000	0x3FFFFFFF	-			Reserved
System RAM—see Table 5-5 for details					
0x40000000	0x4005FFFF	384 KB		-	System RAM
0x40060000	0x4007FFFF	-			Reserved System RAM
0x40080000	0x4FFFFFFF	-		-	Reserved
Local memory					
0x50000000	0x5000FFFF	-	-	-	Reserved
0x50010000	0x507FFFFF	-	-		Reserved
0x50800000	0x5080FFFF	64 KB	64 KB	-	D-MEM CPU0
0x50810000	0xF7FFFFFF	-	-		Reserved
Peripherals PBRIDGE_1					
0xF8000000	0xF8003FFF	16 KB		-	Peripheral Bridge 1 (PBRIDGE_1)
0xF8004000	0xF87FFFFF	-			Reserved
0xFBC00000	0xFBC03FFF	16 KB		255	FlexPWM 0 (FlexPWM_0
0xFBC04000	0xFBC0FFFF	-			Reserved
0xFBC10000	0xFBC13FFF	16 KB		251	CTU 0 (CTU_0)
0xFBC14000	0xFBC1FFFF	-			Reserved
0xFBC20000	0xFBC23FFF	16 KB		247	ETIMER 0 (ETIMER_0)
0xFBC24000	0xFBC27FFF	-			Reserved
0xFBC28000	0xFBC2BFFF	16 KB		245	ETIMER 2 (ETIMER_2)
0xFBC2C000	0xFBC2FFFF	-			Reserved
0xFBC40000	0xFBC43FFF	16 KB		239	SGEN 0 (SGEN_0)
0xFBBE4000	0xFBDFFFFF	-			Reserved
0xFBE00000	0xFBE03FFF	16 KB		237	SAR ADC 0 (ADC_0)
0xFBE04000	0xFBE07FFF	16 KB			Reserved
0xFBE08000	0xFBE0BFFF	16 KB		235	SAR ADC 2 (ADC_2)
0xFBE0C000	0xFBE5BFFF	-			Reserved
0xFBE5C000	0xFBE5FFFF	16 KB		214	SENT Receiver 1 (SRX_1)
0xFBE60000	0xFBE6FFFF	-			Reserved
0xFBE70000	0xFBE73FFF	16 KB		209	Deserial Serial Peripheral Interface 2 (DSPI_2)

Table continues on the next page...

Table 5-1. Overview memory map (continued)

Start address	End address	Allocated size	Used size	PCTL number	Description
0xFBE74000	0xFBE77FFF	16 KB		208	Deserial Serial Peripheral Interface 3 (DSPI_3)
0xFBE78000	0xFBE83FFF	-			Reserved
0xFBE84000	0xFBE87FFF	16 KB		204	LIN Controller 0 (LINFlex_0)
0xFBE88000	0xFBF57FFF	-			Reserved
0xFBF58000	0xFBF5BFFF	16 KB		-	Fault Collection and Control Unit (FCCU)
0xFBF5C000	0xFBF6BFFF	-			Reserved
0xFBF6C000	0xFBF6FFFF	16 KB		146	Direct Memory Access Multiplexer (DMAMUX_1)
0xFBF70000	0xFBFB41BF	-			Reserved
0xFBFB0200	0xFBFB023F	64 bytes		-	Clock Monitor Unit for Motor Clock (CMU_0)
0xFBFB0240	0xFBFB027F	64 bytes		-	Clock Monitor Unit for SYS_CLK (CMU_1)
0xFBFB0280	0xFBFB02BF	64 bytes		-	Clock Monitor Unit for Peripheral Bridge (CMU_2)
0xFBFB02C0	0xFBFB02FF	64 bytes		-	Clock Monitor Unit for ADC Clock (CMU_3)
0xFBFB0300	0xFBFB03FF	64 bytes		-	Clock Monitor Unit for SENT (CMU_4)
0xFBFB0340	0xFBFFFFFF	-			Reserved
Peripherals PBRIDGE_0					
0xFC000000	0xFC003FFF	16 KB		-	Peripheral Bridge 0 (PBRIDGE_0)
0xFC004000	0xFC007FFF	16 KB		-	Crossbar 0 (XBAR_0)
0xFC008000	0xFC00FFFF	-			Reserved
0xFC010000	0xFC013FFF	16 KB		-	System Memory Protection Unit 0 (SMPU_0)
0xFC014000	0xFC017FFF	-			Reserved
0xFC018000	0xFC01BFFF	16 KB		-	Crossbar Integrity Checker 0 (XBIC_0)
0xFC01C000	0xFC01FFFF	-			Reserved
0xFC020000	0xFC023FFF	16 KB		-	Platform RAM controller (PRAM)

Table continues on the next page...

Table 5-1. Overview memory map (continued)

Start address	End address	Allocated size	Used size	PCTL number	Description
0xFC024000	0xFC027FFF	-			Reserved
0xFC028000	0xFC02BFFF	16 KB		-	Platform control module (PCM)
0xFC02C000	0xFC02FFFF	16 KB		-	Reserved
0xFC030000	0xFC033FFF	16 KB		-	Platform Flash controller (PFLASH)
0xFC034000	0xFC037FFF	16 KB		-	Crossbar Integrity Checker 1 (XBIC_1)
0xFC038000	0xFC03FFFF	-			Reserved
0xFC040000	0xFC043FFF	16 KB		-	Interrupt Controller (INTC_0)
0xFC044000	0xFC04FFFF	-			Reserved
0xFC050000	0xFC053FFF	16 KB		-	Software Watchdog Timer 0 (SWT_0)
0xFC054000	0xFC067FFF	-			Reserved
0xFC068000	0xFC06BFFF	16 KB		-	System Timer Module 0 (STM_0)
0xFC06C000	0xFC07BFFF	-			Reserved
0xFC07C000	0xFC07FFFF	16 KB		-	Error Injection Module (EIM)
0xFC080000	0xFC09FFFF	-			Reserved
0xFC0A0000	0xFC0A3FFF	16 KB		-	Direct Memory Access Controller 0 (DMA_0)
0xFC0A4000	0xFC0AFFFF	-			Reserved
0xFC0B0000	0xFC0B3FFF	16 KB		12	Ethernet 0 (ENET_0)
0xFC0B4000	0xFFC00103	-			Reserved
0xFFC00104	0xFFC00107				DCL_IPS0 register (used by STCU2)
0xFFC00108	0xFFC03FFF	-			Reserved
0xFFC04000	0xFFC07FFF	16 KB		144	FlexPWM 1 (FlexPWM_1)
0xFFC08000	0xFFC0BFFF	-			Reserved
0xFFC0C000	0xFFC0C003	-			Generic Control Register
0xFFC0C004	0xFFC0C007	-			RCCU_STAT register
0xFFC0C008	0xFFC13FFF	-			Reserved
0xFFC14000	0xFFC17FFF	16 KB		141	CTU 1 (CTU_1)
0xFFC18000	0xFFC1FFFF	-			Reserved

Table continues on the next page...

Table 5-1. Overview memory map (continued)

Start address	End address	Allocated size	Used size	PCTL number	Description
0xFFC24000	0xFFC27FFF	16 KB		137	ETIMER 1 (ETIMER_1)
0xFFC28000	0xFFE03FFF	-			Reserved
0xFFE04000	0xFFE07FFF	16 KB		126	SAR ADC 1 (ADC_1)
0xFFE08000	0xFFE0BFFF	-			Reserved
0xFFE0C000	0xFFE0FFFF	16 KB		124	SAR ADC 3 (ADC_3)
0xFFE10000	0xFFE4FFFF	-			Reserved
0xFFE50000	0xFFE53FFF	16 KB		107	FlexRay Communication Controller 0 (FLEXRAY_0)
0xFFE54000	0xFFE5BFFF	-			Reserved
0xFFE5C000	0xFFE5FFFF	16 KB		104	SENT Receiver 0 (SRX_0)
0xFFE60000	0xFFE6FFFF	-			Reserved
0xFFE70000	0xFFE73FFF	16 KB		99	Deserial Serial Peripheral Interface 0 (DSPI_0)
0xFFE74000	0xFFE77FFF	16 KB		98	Deserial Serial Peripheral Interface 1 (DSPI_1)
0xFFE78000	0xFFE8FFFF	-			Reserved
0xFFE90000	0xFFE93FFF	16 KB		91	LIN Controller 1 (LINFlex_1)
0xFFE94000	0xFFEDFFFF	-			Reserved
0xFFEC0000	0xFFEC3FFF	16 KB		79	FlexCAN 0 (CAN_0)
0xFFEC4000	0xFFEC7FFF	16 KB		78	FlexCAN 1 (CAN_1)
0xFFEC8000	0xFFECBFFF	16 KB		77	FlexCAN 2 (CAN_2)
0xFFEEC000	0xFFF43FFF	-			Reserved
0xFFF44000	0xFFF47FFF	16 KB		-	Self Test Control Unit (STCU2)
0xFFF48000	0xFFF4FFFF	-			Reserved
0xFFF50000	0xFFF53FFF	16 KB		-	Memory Error Management Unit (MEMU)
0xFFF54000	0xFFF57FFF	16 KB			Reserved
0xFFF58000	0xFFF63FFF	-			Reserved
0xFFF64000	0xFFF67FFF	16 KB		38	Cyclic Redundancy Check 0 (CRC_0)
0xFFF68000	0xFFF6BFFF	-			Reserved

Table continues on the next page...

Table 5-1. Overview memory map (continued)

Start address	End address	Allocated size	Used size	PCTL number	Description
0xFFF6C000	0xFFF6FFFF	16 KB		36	Direct Memory Access Multiplexer (DMAMUX_0)[1]
0xFFF70000	0xFFF73FFF	-			Reserved
0xFFF78000	0xFFF83FFF	-			Reserved
0xFFF84000	0xFFF87FFF	16 KB		30	Periodic Interval Timer 0 (PIT_0)
0xFFF88000	0xFFF97FFF	-			Reserved
0xFFF98000	0xFFF9BFFF	16 KB		-	Wake-Up Unit (WKPU)
0xFFF9C000	0xFFF9FFFF	-			Reserved
0xFFFA0000	0xFFFA3FFF	16 KB[2]		-	Power Control Unit (MC_PCU)
0xFFFA0400	0xFFFA07FF	1 KB		-	Power Management Controller (PMC)
0xFFFA4000	0xFFFA7FFF	-			Reserved
0xFFFA8000	0xFFFABFFF	16 KB		-	Reset Generation Module (MC_RGM)
0xFFFAC000	0xFFFAFFFF	-			Reserved
0xFFFB0000	0xFFFB3FFF	16 KB[3]		-	Clock Generation Module (MC_CGM)
0xFFFB0000	0xFFFB003F	64 bytes		-	Internal RC Oscillator (IRCOSC)
0xFFFB0040	0xFFFB007F	-			Reserved
0xFFFB0080	0xFFFB00BF	64 bytes		-	Oscillator (XOSC)
0xFFFB00C0	0xFFFB00FF	-			Reserved
0xFFFB0100	0xFFFB013F	64 bytes		-	Dual PLL (PLLDIG)
0xFFFB4000	0xFFFB7FFF	-			Reserved
0xFFFB8000	0xFFFBBFFF	16 KB		-	Mode Entry Module (MC_ME)
0xFFFBC000	0xFFFBFFFF	-			Reserved
0xFFFC0000	0xFFFC3FFF	16 KB		-	System Integration Unit Lite 2 (SIUL2)
0xFFFC4000	0xFFFCFFFF	-			Reserved
0xFFFD0000	0xFFFD3FFF	16 KB		11	Serial Interprocessor Interface 0 (SIPI_0)
0xFFFD4000	0xFFFD7FFF	-			Reserved
0xFFFD8000	0xFFFDBFFF	16 KB		9	LFAST 0 (LFAST_0)
0xFFFDC000	0xFFFDFFFF	-			Reserved

Table continues on the next page...

Table 5-1. Overview memory map (continued)

Start address	End address	Allocated size	Used size	PCTL number	Description
0xFFFE0000	0xFFFE3FFF	16 KB		-	Flash main control registers
0xFFFE4000	0xFFFF7FFF	-			Reserved
0xFFFF8000	0xFFFFBFFF	16 KB		-	System Status and Configuration Module (SSCM)
0xFFFFC000	0xFFFFFFFF	16 KB		-	Boot Assist Module (BAM)

1. DMA is a platform IP and in low power modes the clock to all platform IPs is shut off. Therefore the DMA cannot be kept ON in low power modes. So PCTL has no impact during STOP/HALT. modes.
2. The PMC register locations overlap with the memory map slot of the MC_PCU. For more information, see Power control registers.
3. The locations of the IRCOSC, XOSC, and PLLDIG modules' registers overlap with the memory map slot of the MC_CGM. For more information, see Clock generation registers.

5.1.1 Power control registers

The MC_PCU works as a bus bridge for the PMC module. The locations of the PMC's registers overlap with the memory map slot allocated to the MC_PCU. The following table identifies the offsets of the MC_PCU and PMC registers relative to the base address of the MC_PCU.

Table 5-2. MC_PCU memory slot and relative register locations

Start offset (hex)	End offset (hex)	Module registers
0000	0043	MC_PCU registers
0044	03FF	Reserved
0400	07FF	PMC registers
0800	3FFF	Reserved

5.1.2 Clock generation registers

The MC_CGM works as a bus bridge for the IRCOSC, XOSC, and PLLDIG modules. The locations of these modules' registers overlap with the memory map slot allocated to the MC_CGM. The following table identifies the offsets of these registers, including the actual MC_CGM registers, relative to the base address of the MC_CGM.

Table 5-3. MC_CGM memory slot and relative module register locations

Start offset (hex)	End offset (hex)	Module registers
0000	003F	IRCOSC registers
0040	007F	Reserved
0080	00BF	XOSC registers
00C0	00FF	Reserved
0100	013F	PLLDIG registers
0140	06FF	Reserved
0700	08CB	MC_CGM registers
08CC	3FFF	Reserved

5.1.3 Flash memory and RAM memory maps

The following table is the detailed flash memory map for the MPC5744P.

Table 5-4. Flash memory map

Start address	End address	Allocated size (KB)	Complete flash memory block structure	RWW partition	Block size
Reserved - no overlay					
0x00000000	0x003FFFFF	-	Reserved		
UTest NVM block - no overlay					
0x00400000	0x00403FFF	16	UTest NVM block space 16 KB		16 KB
0x00404000	0x007FFFFF	-	Reserved		
Data flash memory - no overlay					
0x00800000	0x00803FFF	16	EEPROM - low block 0	0	16 KB
0x00804000	0x00807FFF	16	EEPROM - low block 1	0	16 KB
0x00808000	0x0080FFFF	32	EEPROM - mid block 0	2	32 KB
0x00810000	0x00817FFF	32	EEPROM - mid block 1	3	32 KB
0x00818000	0x009FFFFF	-	Reserved		
Small flash memory blocks - no overlay					
0x00A00000	0x00F97FFF	-	Reserved		
0x00F98000	0x00F9BFFF	16	16 KB low flash memory block 2 (boot location 0)	1	16 KB

Table continues on the next page...

Table 5-4. Flash memory map (continued)

Start address	End address	Allocated size (KB)	Complete flash memory block structure	RWW partition	Block size
0x00F9C000	0x00F9FFFF	16	16 KB low flash memory block 3 (boot location 1)	1	16 KB
Medium flash memory blocks - no overlay					
0x00FA0000	0x00FAFFFF	64	64 KB high flash memory block 0 (boot location 2)	4	64 KB
0x00FB0000	0x00FBFFFF	64	64 KB high flash memory block 1 (boot location 3)	4	64 KB
0x00FC0000	0x00FCFFFF	64	64 KB high flash memory block 2	4	64 KB
0x00FD0000	0x00FDFFFF	64	64 KB high flash memory block 3	5	64 KB
0x00FE0000	0x00FEFFFF	64	64 KB high flash memory block 4	5	64 KB
0x00FF0000	0x00FFFFFF	64	64 KB high flash memory block 5	5	64 KB
Large flash memory blocks - no overlay					
0x01000000	0x0103FFFF	256	256 KB flash memory block 0 (boot location 4)	6	256 KB
0x01040000	0x0107FFFF	256	256 KB flash memory block 1 (boot location 5)	6	256 KB
0x01080000	0x010BFFFF	256	256 KB flash memory block 2 (boot location 6)	6	256 KB
0x010C0000	0x010FFFFF	256	256 KB flash memory block 3 (boot location 7)[1]	6	256 KB
0x01100000	0x0113FFFF	256	256 KB flash memory block 4	7	256 KB
0x01140000	0x0117FFFF	256	256 KB flash memory block 5[2]	7	256 KB
0x01180000	0x011BFFFF	256	256 KB flash memory block 6	7	256 KB
0x011C0000	0x011FFFFF	256	256 KB flash memory block 7[3]	7	256 KB
0x01200000	0x01FFFFFF	-	Reserved		
Reserved flash memory - no overlay					
0x02000000	0x07FFFFFF	-	Reserved		
Mirror reserved flash memory					
0x08000000	0x083FFFFF	-	Reserved		

Table continues on the next page...

Table 5-4. Flash memory map (continued)

Start address	End address	Allocated size (KB)	Complete flash memory block structure	RWW partition	Block size
Mirror reserved flash memory					
0x08400000	0x089FFFFF	-	Reserved		
Mirror small flash memory blocks - no overlay					
0x08A00000	0x08F97FFF	-	Reserved		
0x08F98000	0x08F9BFFF	16	16 KB low flash memory block 2	1	16 KB
0x08F9C000	0x08F9FFFF	16	16 KB low flash memory block 3	1	16 KB
Mirror medium flash memory blocks - no overlay					
0x08FA0000	0x08FAFFFF	64	64 KB high flash memory block 0	4	64 KB
0x08FB0000	0x08FBFFFF	64	64 KB high flash memory block 1	4	64 KB
0x08FC0000	0x08FCFFFF	64	64 KB high flash memory block 2	4	64 KB
0x08FD0000	0x08FDFFFF	64	64 KB high flash memory block 3	5	64 KB
0x08FE0000	0x08FEFFFF	64	64 KB high flash memory block 4	5	64 KB
0x08FF0000	0x08FFFFFF	64	64 KB high flash memory block 5	5	64 KB
Mirror large flash memory blocks - no overlay					
0x09000000	0x0903FFFF	256	256 KB flash memory block 0	6	256 KB
0x09040000	0x0907FFFF	256	256 KB flash memory block 1	6	256 KB
0x09080000	0x090BFFFF	256	256 KB flash memory block 2	6	256 KB
0x090C0000	0x090FFFFF	256	256 KB flash memory block 3	6	256 KB
0x09100000	0x0913FFFF	256	256 KB flash memory block 4	7	256 KB
0x09140000	0x0917FFFF	256	256 KB flash memory block 5	7	256 KB
0x09180000	0x091BFFFF	256	256 KB flash memory block 6	7	256 KB
0x091C0000	0x091FFFFF	256	256 KB flash memory block 7	7	256 KB
0x09200000	0x09FFFFFF	-	Reserved		
Reserved flash memory					
0x0A000000	0x0FFFFFFF	-	Reserved		

[1] Reserved on MPC5741P

[2] Reserved on MPC5741P and MPC5742P

3. Reserved on MPC5741P, MPC5742P, and MPC5743P

The following table shows the RAM memory map.

Table 5-5. RAM memory map

Start address	End address	Allocated size	Used	Description
System RAM				
0x40000000	0x4001FFFF	128 KB	128 KB	System RAM on MPC5744P, MPC5743P, MPC5742P, and MPC5741P
0x40020000	0x4002FFFF	64 KB	64 KB	System RAM on MPC5744P, MPC5743P, and MPC5742P Reserved on MPC5741P
0x40030000	0x4003FFFF	64 KB	64 KB	System RAM on MPC5744P and MPC5743P Reserved on MPC5742P and MPC5741P
0x40040000	0x4005FFFF	128 KB	128 KB	System RAM on MPC5744P Reserved on MPC5743P, MPC5742P, and MPC5741P
0x40060000	0x4FFFFFFF	-	-	Reserved
Local memory				
0x50000000	0x507FFFFF	-	-	Reserved
0x50800000	0x5080FFFF	64 KB	64 KB	D-MEM CPU0
0x50810000	0x5FFFFFFF	-	-	Reserved

Table 5-6. UTEST flash memory map

Start address	End address	Allocated size (bytes)	Description	Comments
0x00400000	0x0040000B	12	Reserved	
0x0040000C	0x0040000F	4	Reserved	
0x00400010	0x0040001F	16	Reserved	
0x00400020	0x0040002F	16	Reserved	
0x00400030	0x0040003F	16	Reserved	
0x00400040	0x0040005F	32	Customer Single Bit Correction Area	Programmed by factory to include ECC/EDC errors to allow testing of ECC/EDC hardware
0x00400060	0x0040007F	32	Customer Double Bit Detection Area	
0x00400080	0x0040009F	32	Customer EDC after ECC Area	
0x004000A0	0x004000BF	32	Unique Chip Identifier (UID)	Programmed during factory test. The UID includes information on wafer lot, X/Y-position on the wafer, manufacturing data, test results, and company ID (65 for Freescale).

Table continues on the next page...

Table 5-6. UTEST flash memory map (continued)

Start address	End address	Allocated size (bytes)	Description	Comments
0x004000C0	0x004000C7	8	Temperature Sensor 0 calibration	16-bit signed integer calibration parameters K1, K2, K3, and K4 for TSENS_0
0x004000C8	0x004000CF	8	Temperature Sensor 1 calibration	16-bit signed integer calibration parameters K1, K2, K3, and K4 for TSENS_1
0x004000D0	0x004000EB	28	ADC self test threshold values	
0x004000EC	0x004000FF	20	Reserved	
0x00400100	0x00400103	4	Reserved	
0x00400104	0x004001FF	252	Reserved	
0x00400200	0x004009FF	2048	DCF Records	Contiguous list of DCF Records starting at 0x00400200. Initial records programmed by factory. Subsequent records added to the end of the list by the customer.
0x00401000	0x00403FFF	12288	Reserved for customer OTP data	Programmed by the customer

Chapter 6
Functional Safety

6.1 Introduction

The MPC5744P is developed according to ISO 26262 and has an integrated safety concept targeting safety-related systems requiring high safety integrity levels. In order to support the integration of the MPC5744P into safety-related systems, the following documentation is available:

- Reference Manual (MPC5744PRM) - Describes the MPC5744P functionality
- Data Sheet (MPC5744PDS) - Describes the MPC5744P operating conditions
- Safety Manual (MPC5744PSM) - Describes the MPC5744P safety concept and possible safety mechanisms (integrated in MPC5744P, system level hardware or system level software), as well as measures to reduce dependent failures
- Dynamic FMEDA - Inductive analysis enabling customization of system level safety mechanisms, including the resulting safety metrics for ISO 26262 (SPFM, LFM & PMHF) and IEC 61508 (SFF & Beta IC Factor)
- FMEDA Report - Describes the FMEDA methodology and safety mechanisms supported in the FMEDA, including source of failure rates, failure modes and assumptions made during the analysis.

The Reference Manual, Data Sheet and Safety Manual are available for download on the MPC5744P product page. The Dynamic FMEDA and FMEDA report are available upon request. The MPC5744P is a SafeAssure solution; for further information regarding functional safety at Freescale, visit www.freescale.com/safeassure.

Chapter 7
Chip Configuration

7.1 Introduction

This information consists of details about the individual modules of the microcontroller. The information includes:

- module block diagrams showing immediate connections within the device,
- specific module-to-module interactions not necessarily discussed in the individual module chapters, and
- links for more information.

7.2 Core modules

The microcontroller has two separate cores that perform various computational and control functions.

- The Main Core_0 uses an e200z425n3 core. This core is used for general computational functions.
- The Checker Core_0s uses an e200z424 core. When enabled, Checker Core_0s operates in lock step mode with Main Core_0 executing exactly the same instructions as Main Core_0. Thus, Checker Core_0s checks to insure that Main Core_0 is executing correctly.

For more information, see the Core Complex Overview.

7.2.1 Core reset settings

The following table shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately after a system reset.

Table 7-1. Reset settings for e200z4 resources

Resource	System reset setting
Program Counter	p_rstbase[0:29] || 2'b00
GPRs	Unaffected
CR	Unaffected[2]
BUCSR	0x0000_0000
CSRR0	Unaffected[2]
CSRR1	Unaffected[2]
CTR	Unaffected[2]
DAC1	0x0000_0000
DAC2	0x0000_0000[3]
DBCR0	0x0000_0000[3]
DBCR1	0x0000_0000[3]
DBCR2	0x0000_0000[3]
DBCR4	0x0000_0000[3]
DBSR	0x1000_0000[3]
DDAM	0x0000_0000[3]
DDEAR	Unaffected[2]
DEAR	Unaffected[2]
DEVENT	0x0000_0000[3]
DSRR0	Unaffected[2]
DSRR1	Unaffected[2]
DVC1	Unaffected[2]
DVC2	Unaffected[2]
ESR	0x0000_0000
HID0	0x0000_0000
HID1	0x0000_0000
IAC1	0x0000_0000[3]
IAC2	0x0000_0000[3]
IAC3	0x0000_0000[3]
IAC4	0x0000_0000[3]
IAC5	0x0000_0000[3]
IAC6	0x0000_0000[3]
IAC7	0x0000_0000[3]
IAC8	0x0000_0000[3]
IVPR	Unaffected[2]
LR	Unaffected[2]

Table continues on the next page...

Table 7-1. Reset settings for e200z4 resources (continued)

Resource	System reset setting
L1CFG0, L1CFG1	—
L1CSR0, L1CSR1	0x0000_0000
L1FINV0, L1FINV1	0x0000_0000
MAS1	Unaffected[2]
MAS2	Unaffected[2]
MAS3	Unaffected[2]
MAS4	Unaffected[2]
MCAR	Unaffected[2]
MCSR	0x0000_0000
MCSRR0	Unaffected[2]
MCSRR1	Unaffected[2]
MMUCFG[4]	—
MPU0CSR0	0x0000_0000
MPU0CFG[4]	—
MSR	0x0000_0000
NPIDR	0x0000_0000
PID0	0x0000_0000
PIR	0x0000_00 ǁ p_cpuid[0:7][1]
PVR[4]	—
SPEFSCR	0x0000_0000
SPRG0	Unaffected[2]
SPRG1	Unaffected[2]
SPRG2	Unaffected[2]
SPRG3	Unaffected[2]
SRR0	Unaffected[2]
SRR1	Unaffected[2]
SVR[4]	—
XER	0x0000_0000

1. The levels that determine these bits are set by the factory and may change between revisions of the device.
2. Undefined on **POR** assertion, unchanged on **external PORESET (EXT_POR_B)** assertion.
3. Reset by processor reset **external PORESET (EXT_POR_B)** if DBCR0[EDM]=0, as well as unconditionally by **POR**.
4. Read-only registers.

7.2.2 Special Purpose Register summary

PowerISA 2.06 and implementation-specific SPRs for the e200z4251n3 and e200z424 cores are listed in the following table. All registers are 32 bits in size. Register bits are numbered from bit 0 to bit 31 (most-significant to least-significant). Shaded entries represent optional registers. An SPR may be read or written with the **mfspr** and **mtspr** instructions. In the instruction syntax, compilers should recognize the mnemonic name given in the following table.

Table 7-2. Special Purpose Registers

Mnemonic	Name	SPR number	Access	Privileged	e200z specific
XER	Integer Exception Register	1	R/W	No	No
LR	Link Register	8	R/W	No	No
CTR	Count Register	9	R/W	No	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No
SRR1	Save/Restore Register 1	27	R/W	Yes	No
PID0	Process ID Register	48	R/W	Yes	No
CSRR0	Critical Save/Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/Restore Register 1	59	R/W	Yes	No
DEAR	Data Exception Address Register	61	R/W	Yes	No
ESR	Exception Syndrome Register	62	R/W	Yes	No
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
USPRG0 (VRSAVE)	User SPR General 0 (renamed to VRSAVE in PowerISA 2.06)	256	R/W	No	No
SPRG0	SPR General 0	272	R/W	Yes	No
SPRG1	SPR General 1	273	R/W	Yes	No
SPRG2	SPR General 2	274	R/W	Yes	No
SPRG3	SPR General 3	275	R/W	Yes	No
PIR	Processor ID Register	286	R/W	Yes	No
PVR	Processor Version Register	287	Read-only	Yes	No
DBSR	Debug Status Register	304	Read/Clear[1]	Yes	No
DBCR0	Debug Control Register 0	308	R/W	Yes	No
DBCR1	Debug Control Register 1	309	R/W	Yes	No
DBCR2	Debug Control Register 2	310	R/W	Yes	No
IAC1	Instruction Address Compare 1	312	R/W	Yes	No
IAC2	Instruction Address Compare 2	313	R/W	Yes	No
IAC3	Instruction Address Compare 3	314	R/W	Yes	No
IAC4	Instruction Address Compare 4	315	R/W	Yes	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	Yes	No
DVC1	Data Value Compare 1	318	R/W	Yes	No
DVC2	Data Value Compare 2[2]	319	R/W	Yes	No
TIR	Thread identification register	446	Read-only	Yes	No
SPEFSCR	LSP/EFP APU status and control register	512	R/W	No	No
L1CFG0	L1 cache config register 0	515	Read-only	No	Yes
L1CFG1	L1 cache config register 1	516	Read-only	No	Yes

Table continues on the next page...

Table 7-2. Special Purpose Registers (continued)

Mnemonic	Name	SPR number	Access	Privileged	e200z specific
NPIDR	Nexus 3 Process ID register	517	R/W	No	Yes
DBCR3	Debug control register 3	561	R/W	Yes	Yes
DBCR4	Debug control register 4	563	R/W	Yes	Yes
DBCR5	Debug control register 5	564	R/W	Yes	Yes
IAC5	Instruction Address Compare 5	565	R/W	Yes	Yes
IAC6	Instruction Address Compare 6	566	R/W	Yes	Yes
IAC7	Instruction Address Compare 7	567	R/W	Yes	Yes
IAC8	Instruction Address Compare 8	568	R/W	Yes	Yes
MCSRR0	Machine Check Save/Restore Register 0	570	R/W	Yes	Yes
MCSRR1	Machine Check Save/Restore Register 1	571	R/W	Yes	Yes
MCSR	Machine Check Syndrome Register	572	Read/Clear[3]	Yes	Yes
MCAR	Machine Check Address Register	573	R/W	Yes	Yes
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
DDAM	Debug Data Acquisition Messaging register	576	R/W	No	Yes
DAC3	Data Address Compare 3	592	R/W	Yes	Yes
DAC4	Data Address Compare 4	593	R/W	Yes	Yes
DBCR7	Debug control register 7	596	R/W	Yes	Yes
DBCR8	Debug control register 8	597	R/W	Yes	Yes
DDEAR	Debug Data Effective Address register	600	R/W	Yes	Yes
DVC1U	Data Value Compare 1 Upper	601	R/W	Yes	No
DVC2U[4]	Data Value Compare 2 Upper	602	R/W	Yes	No
DBCR6	Debug control register 6	603	R/W	Yes	Yes
MAS0	MPU assist register 0	624	R/W	Yes	Yes
MAS1	MPU assist register 1	625	R/W	Yes	Yes
MAS2	MPU assist register 2	626	R/W	Yes	Yes
MAS3	MPU assist register 3	627	R/W	Yes	Yes
EDBRAC0	External debug resource allocation control register 0	638	Read only	Yes	Yes
MPU0CFG	MPU0 configuration register	692	Read-only	Yes	Yes
DMEMCFG0	Local Data Memory config register 0	694	Read-only	Yes	Yes
L1FINV1	L1 cache flush and invalidate control register 0	959	R/W	Yes	Yes
DEVENT	Debug Event register	975	R/W	No	Yes
SIR	System Information Register	992	Read-only	No	Yes
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
L1CSR0	L1 cache control and status register 0	1010	R/W	Yes	Yes
L1CSR1	L1 cache control and status register 1	1011	R/W	Yes	Yes
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes

Table continues on the next page...

Table 7-2. Special Purpose Registers (continued)

Mnemonic	Name	SPR number	Access	Privileged	e200z specific
MPU0CSR0	MPU0 configuration register	1014	R/W	Yes	Yes
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
L1FINV0	L1 cache flush and invalidate control register 0	1016	R/W	Yes	Yes
SVR	System Version Register	1023	Read-only	Yes	Yes

1. The Debug Status Register can be read using *mfspr RT,DBSR*. The Debug Status Register cannot be written to directly. Instead, bits in the Debug Status Register corresponding to ‘1’ bits in GPR(RS) can be cleared using *mtspr DBSR,RS*.
2. Undefined on **POR** assertion, unchanged on **external PORESET (EXT_POR_B)** assertion.
3. The Machine Check Syndrome Register can be read using *mfspr RT,MCSR*. The Machine Check Syndrome Register cannot be written to directly. Instead, bits in the Machine Check Syndrome Register corresponding to ‘1’ bits in GPR(RS) can be cleared using *mtspr MCSR,RS*.
4. The 64-bit DVC registers are accessed by using the DVC1 and DVC2 SPR numbers for the lower 32 bits and the DVC1U and DVC2U SPR numbers for the upper 32 bits.

7.2.3 Core reservation instructions

This device’s core platform does not use reservation functionality. The core executes reservation instructions, but these instructions do not block other bus masters.

7.3 Platform Configuration Module (PCM)

The Platform Configuration Module contains miscellaneous configuration registers for the device. Currently, the configuration registers are related to the operation of the intelligent bus bridging gasket. The module is mapped to AIPS on-platform slot 10 with a base address of 0xFC02_8000.

PCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)	32	R/W	0007_0707h	7.3.1/179
8	Bus Bridge Configuration Register 2 (PCM_IAHB_BE2)	32	R/W	0007_0707h	7.3.2/180
C	Bus Bridge Configuration Register 3 (PCM_IAHB_BE3)	32	R/W	0000_0707h	7.3.3/181

7.3.1 Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)

This register, which must be accessed in supervisor mode, configures the bus bridge gasket for the DMA/SIPI/Ethernet port concentrator.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					Reserved			0					Reserved	BRE	BWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					Reserved			0					Reserved		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

PCM_IAHB_BE1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 Reserved	This field is reserved.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved.
14 BRE	Burst Read Enable Controls the bus gasket's handling of burst read transactions for the DMA/SIPI/Ethernet port concentrator. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
15 BWE	Burst Write Enable Controls the bus gasket's handling of burst write transactions for the DMA/SIPI/Ethernet port concentrator. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PCM_IAHB_BE1 field descriptions (continued)

Field	Description
21–23 Reserved	This field is reserved.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved.

7.3.2 Bus Bridge Configuration Register 2 (PCM_IAHB_BE2)

This register, which must be accessed in supervisor mode, configures the bus bridge gasket for the FlexRay module.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													Reserved		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					Reserved	BRE	BWE	0					Reserved		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

PCM_IAHB_BE2 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 Reserved	This field is reserved.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved.
22 BRE	Burst Read Enable Controls the bus gasket's handling of burst read transactions for the FlexRay module.

Table continues on the next page...

PCM_IAHB_BE2 field descriptions (continued)

Field	Description
	0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
23 BWE	Burst Write Enable Controls the bus gasket's handling of burst write transactions for the FlexRay module. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved.

7.3.3 Bus Bridge Configuration Register 3 (PCM_IAHB_BE3)

This register, which must be accessed in supervisor mode, configures the bus bridge gasket for PBRIDGE_0 and PBRIDGE_1.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					PRE_B	BRE_B	BWE_B	0					PRE_A	BRE_A	BWE_A
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

PCM_IAHB_BE3 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_B	Pending Read Enable Controls the bus gasket's handling of pending read transactions for PBRIDGE_1.

Table continues on the next page...

PCM_IAHB_BE3 field descriptions (continued)

Field	Description
	0 Pending reads are disabled. 1 Pending writes are enabled.
22 BRE_B	Burst Read Enable Controls the bus gasket's handling of burst read transactions for PBRIDGE_1. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
23 BWE_B	Burst Write Enable Controls the bus gasket's handling of burst write transactions for PBRIDGE_1. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_A	Pending Read Enable Controls the bus gasket's handling of pending read transactions for PBRIDGE_0. 0 Pending reads are disabled. 1 Pending writes are enabled.
30 BRE_A	Burst Read Enable Controls the bus gasket's handling of burst read transactions for PBRIDGE_0. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
31 BWE_A	Burst Write Enable Controls the bus gasket's handling of burst write transactions for PBRIDGE_0. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

7.4 System modules

7.4.1 System Integration Unit Lite2 (SIUL2) configuration

The SIUL2 controls MCU reset configuration, pad configuration, external interrupt, general purpose I/O (GPIO), internal peripheral multiplexing, and the system reset operation. The reset configuration block contains the external pin boot configuration logic. The pad configuration block controls the static electrical characteristics of I/O pins. The GPIO block provides uniform and discrete input/output control of the I/O pins of the MCU.

The SIUL2 provides the following features:

- Centralized pad control on per pin basis
- Pin function selection
- Configurable weak pull-up/down
- Configurable slew rate control
- Hysteresis on GPIO pins
- Configurable automatic safe mode pad control
- Input filtering for external interrupts with digital glitch filters

7.4.1.1 SIUL2 interrupts

The SIUL2 supports 32 external interrupts on this device.

The 32 external interrupt sources are grouped into 4 sets with 8 sources each. Each of the 4 groups is assigned one interrupt vector and mapped to the INTC. Each external interrupt source can be masked, and its status can be read individually. Glitch filters are implemented for each external interrupt source.

7.4.1.2 Reset values of MIDRn

The following table provides these registers' reset values.

Table 7-3. MIDRn reset values

Register		144LQFP	257MAPBGA
MIDR1[1]	1N15P	5744_3413h	5744_4013h[2]
	1N65H	5744_3412h	5744_4012h
MIDR2		3900_5000h for MPC5744 3800_5000h for MPC5743 3200_5000h for MPC5742 3000_5000h for MPC5741	3900_5000h for MPC5744 3800_5000h for MPC5743 3200_5000h for MPC5742 3000_5000h for MPC5741

1. The 5744h portion of the reset value applies for MPC5744P, MPC5743P, MPC5742P, and MPC5741P.
2. The reset value of the PKG field corresponds to a 208MAPBGA package because the 257MAPBGA package has 208 balls in its outer sections. The 257MAPBGA package has an additional 49 power and ground balls in an inner section.

7.4.2 Crossbar Switch Integrity Checker (XBIC) configuration

XBIC_0 on this device supports the master and slave ports shown in Table 7-4.

Table 7-4. XBIC_0 master and slave ports supported on this chip

Port type	Available port numbers
Master	0, 1, 5, 6
Slave	0, 2, 4, 5, 7

As a result, for XBIC_0, the available fields in the MCR and ESR are as shown in Table 7-5. The other fields are unused.

Table 7-5. XBIC_0 MCR and ESR field availability on this chip

Register	Available fields
MCR	ME0 ME1 ME5 ME6 SE0 SE2 SE4 SE5 SE7
ESR	DPME0 DPME1 DPME5 DPME6 DPSE0 DPSE2 DPSE4 DPSE5 DPSE7 MST SLV SYN VLD

XBIC_1 supports the three masters connected to the 3-to-1 port concentrator.

Table 7-6. XBIC_1 connections

XBIC_1 port	Connects to
Master 2 (M2)	ENET
Master 1 (M1)	SIPI
Master 0 (M0)	DMA_0
Slave 0 (S0)	AXBS module's M5 port

As a result, for XBIC_1, the available fields in the MCR and ESR are as shown in Table 7-7. The other fields are unused.

Table 7-7. XBIC_1 MCR and ESR field availability on this chip

Register	Available fields
MCR	ME0 ME1 ME2 SE0
ESR	DPME0 DPME1 DPME2 DPSE0 MST SLV SYN VLD

7.4.3 Crossbar Switch (XBAR) configuration

The multi-port Crossbar Switch concurrently supports up to 4 simultaneous connections between master ports and slave ports. It supports the AMBA AHB2.0 AHB-Lite protocol with AMBA V6 extensions for exclusive access support, extended cache control attributes, and misaligned data transfers (referred to as AHB2v6). The Crossbar Switch supports a 32-bit address bus width and a 32 or 64-bit data bus width at all master and slave ports. It also supports both address and data sideband signals which are used to implement decorated storage and the e2eECC for the MPC5744P. A flow-through design allows zero wait-state slave responses.

The Crossbar Switch provides the following features:

- Four master ports and five slave ports (see the following figure)

- 32-bit Address, 64-bit Data paths (applies to all ports) with misaligned access signaling
- Concurrent transfers between independent master and slave ports
- Programmable arbitration priorities on a per-slave port basis
- Round-robin arbitration available on a per-slave port basis
- Parking on slave ports: explicit master, park_on_last_master, none (low power parking)

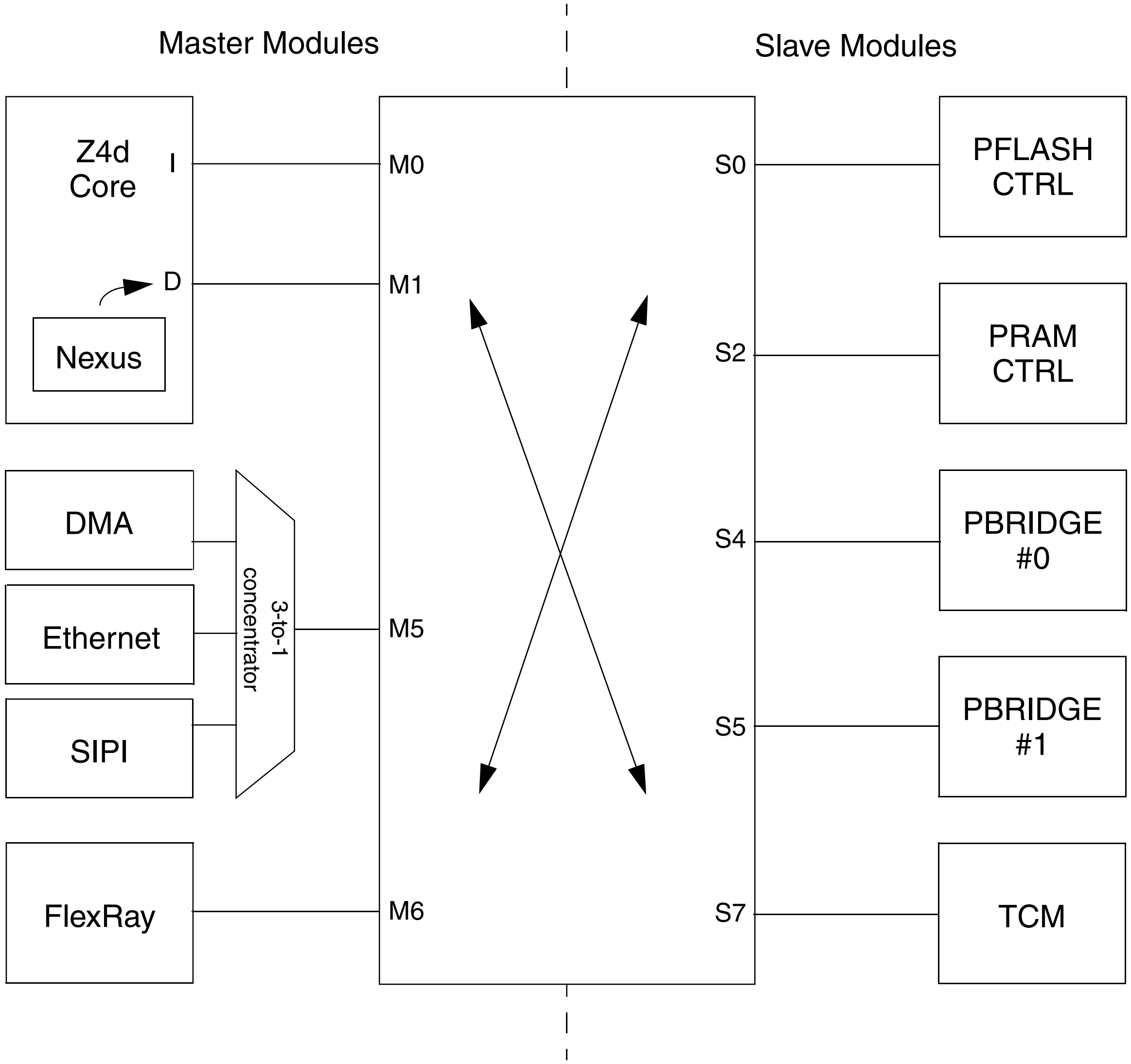


Figure 7-1. Crossbar Switch Integration

7.4.3.1 High priority requests

Each slave on the Crossbar Switch has a CRS register with an HPE field for each master. When a particular HPE field is 1, the Crossbar Switch allows the corresponding master to send high priority requests.

However, not all masters are capable of enabling high priority requests. On this device, FlexRay is the only master that enables high priority requests. The modules that are assigned to the other physical master ports have CRSx[HPEn] fields, but those modules are not capable of enabling high priority requests.

7.4.3.2 Unimplemented PRS and CRS registers

These registers are not implemented because the corresponding slave ports (1, 3, and 6) of the Crossbar Switch are not connected:.

- PRS1, PRS3, and PRS6
- CRS1, CRS3, and CRS6

7.4.3.3 Reset value of PRS and CRS registers

The reset value of the Crossbar Switch's PRS registers is 0320_0010h.

The reset value of the Crossbar Switch's CRS registers is 0000_0000h.

7.4.3.4 Logical master IDs

The following table defines the logical master IDs used by the bus masters connected to the Crossbar Switch.

Table 7-8. Logical master IDs

Master	Logical master ID
z4d_0 core complex Instruction port z4d_0 core complex Load/Store port	0
DMA	2
FlexRay	3
SIPI	4
Ethernet	5
z4d_0 core Nexus	8

7.4.3.5 Port allocation

The following table defines the master port allocation. The ports M2, M3, M4, and M7 are not connected.

Table 7-9. Crossbar Switch master port allocation

Master port	Connected master	Comments
M0	z4d_0 core complex Instruction port	
M1	z4d_0 core complex Load/Store port z4d_0 core Nexus port	Nexus also uses the Load/Store port
M5	DMA/SIPI/Ethernet	Output of the 3-to-1 concentrator
M6	FlexRay	

The following table defines the slave port connections. The ports S1, S3, and S6 are not connected.

Table 7-10. Crossbar Switch slave port allocation

Slave port	Connected slave	Comments
S0	PFLASHC	Platform Flash Memory Controller
S2	PRAMC	Platform SRAM Controller
S4	PBRIDGE0	Peripheral Bridge #0
S5	PBRIDGE1	Peripheral Bridge #1
S7	Data Local Memory	Backdoor access to Data Tightly Coupled Memory (Data TCM)

7.4.4 System Memory Protection Unit (SMPU) configuration

Memory protection isolates user and supervisor tasks through region descriptors that define access control rights. The device provides three layers of memory protection:

1. Processor core memory protection (CMPU) in the core, typically for the core's local memories (DMEM)
2. **System Memory Protection Unit (SMPU)** on the slave side of the crossbar switch
3. Peripheral access control registers (PACRs) for each address space slot in the PBRIDGE

For the second layer of memory protection, this device contains one instance of the SMPU.

The SMPU splits the physical memory into 16 different regions. Each master can be assigned different access rights to each region.

- 16 region MPU with concurrent checks against each master access
- 32 byte granularity for protected address region

See Logical master IDs for master assignments and System Memory Protection Unit (SMPU) for a detailed description of the module.

7.4.5 Peripheral Bridge configuration

The peripheral bridge (PBRIDGE or AIPS) modules are used to access the registers of most of the modules on this device.

7.4.5.1 Number of peripheral bridges

This device contains two identical peripheral bridge instances.

7.4.5.2 Memory maps

See the Memory Map for the memory slot assignment for each module.

7.4.5.3 MPRA register

Each Peripheral Bridge's MPRA register contains fields for each Crossbar Switch master on the chip.

The fields of the MPRA register apply to logical master IDs, not physical port numbers. The settings of MPRA's MTR0, MTW0, and MPL0 fields apply to both logical master 0 and logical master 8.

See Logical master IDs for the master ID assignments on this chip.

7.4.5.4 PACR/OPACR registers

The peripherals attached to the peripheral bridges each are assigned to a memory map slot that corresponds to a peripheral bridge register field. Every on-platform peripheral has an assigned PACRn field within the PACRA to PACRH registers, and every off-platform peripheral has an assigned OPACRn field within the OPACRA to OPACRAF registers.

The following tables provide the peripheral slot assignments for this device. Unused PACRn fields are reserved.

Table 7-11. On-platform peripherals: PBRIDGE_1

Peripheral	PACR
PBRIDGE_1	0

Table 7-12. Off-platform peripherals: PBRIDGE_1

Peripheral	OPACR
FlexPWM 0	255
CTU 0	251
ETIMER 0	247
ETIMER 2	245
SGEN 0	239
SAR ADC 0	127
SAR ADC 2	125
SENT Receiver (SENT 1)	104
Deserial Serial Peripheral Interface 2	99
Deserial Serial Peripheral Interface 3	98
LIN Controller 0	94
Fault Collection and Control Unit	41
Direct Memory Access Multiplexer 1	36
Clock Monitor Unit for motor control clock	18
Clock Monitor Unit for SYS_CLK	18
Clock Monitor Unit for Peripheral Bridge	18
Clock Monitor Unit for ADC clock	18
Clock Monitor Unit for SENT	18

Table 7-13. On-platform peripherals: PBRIDGE_0

Peripheral	PACR
PBRIDGE_0	0
Crossbar 0	1
System Memory Protection Unit 0	4
XBIC_0	6
Platform RAM Controller	8
Platform Control Module	10
Reserved	11
Platform Flash Controller	12
XBIC_1	13
Interrupt Controller 0	16

Table continues on the next page...

Table 7-13. On-platform peripherals: PBRIDGE_0 (continued)

Peripheral	PACR
Software Watchdog Timer 0	20
System Timer Module 0	26
Direct Memory Access Controller 0	40

Table 7-14. Off-platform peripherals: PBRIDGE_0

Peripheral	OPACR
FlexPWM 1	254
CTU 1	250
ETIMER 1	246
SAR ADC 1	126
SAR ADC 3	124
FlexRay Communication Controller	107
SENT Receiver (SENT 0)	104
Deserial Serial Peripheral Interface 0	99
Deserial Serial Peripheral Interface 1	98
LIN Controller 1	91
FlexCAN 0	79
FlexCAN 1	78
FlexCAN 2	77
Self Test Controller Unit	46
Memory Error Management Unit	43
Cyclic Redundancy Check 0	38
Direct Memory Access Multiplexer	36
Periodic Interval Timer 0	30
Wake-Up Unit	25
Power Control Unit (MC_PCU)	23
Power Management Controller (PMC)	22
Reset Generation Module (MC_RGM)	21
Clock Generation Module (MC_CGM)	19
XOSC	19
Dual PLL (PLLDIG)	19
Mode Entry Module (MC_ME)	17
System Integration Unit (SIUL2)	15
Ethernet (ENET)	12
Serial Interprocessor Interface 0	11
LFAST 0	9
Flash memory main control registers	7
System Status and Configuration Module	1
Boot Assist Module	0

7.4.5.5 PBRIDGE register reset values

The following table identifies reset values of PBRIDGE registers.

Table 7-15. PBRIDGE register reset values

Register	PBRIDGE A	PBRIDGE B
MPRA	0x7000_0000	0x7000_0000
PACRA	0x5400_4040	0x5000_0000
PACRB	0x4044_4400	0x0000_0000
PACRC	0x4000_4000	0x0000_0000
PACRD	0x0040_0004	0x0000_0000
PACRF	0x4000_4000	0x0000_0000
PACRG	0x0000_0000	0x0000_0000
PACRH	0x0000_0000	0x0000_0000
OPACRA (OPACR0 to OPACR7) to OPACRAF (OPACR248 to OPACR255)	0x4444_4444	0x4444_4444

7.4.6 Interrupt Controller (INTC) configuration

This device contains one Interrupt Controller (INTC).

7.4.6.1 INTC number of processors and monitors

7.4.6.1.1 Number of processors

The INTC on this chip supports one processor. It is designated processor 0 in the dedicated INTC chapter.

Any information about other processor core numbers in the INTC's chapter does not apply. For example, INTC register details for other processor core numbers are reserved on this chip.

7.4.6.1.2 Interrupt Controller Monitor (INTCM)

The INTC on this chip supports one monitor because the chip has only one core. The INTCM monitors three interrupts.

7.4.6.2 Interrupt vector assignments

The following table defines the interrupt sources for the INTC on the device. All IRQ numbers that are not specifically listed are not used.

Table 7-16. Interrupt vector table

Vector	Source signal	Source module
	Section A: on-platform	
0	Software settable flag 0	INTC (Software)
1	Software settable flag 1	INTC (Software)
2	Software settable flag 2	INTC (Software)
3	Software settable flag 3	INTC (Software)
4	Software settable flag 4	INTC (Software)
5	Software settable flag 5	INTC (Software)
6	Software settable flag 6	INTC (Software)
7	Software settable flag 7	INTC (Software)
8	Software settable flag 8	INTC (Software)
9	Software settable flag 9	INTC (Software)
10	Software settable flag 10	INTC (Software)
11	Software settable flag 11	INTC (Software)
12	Software settable flag 12	INTC (Software)
13	Software settable flag 13	INTC (Software)
14	Software settable flag 14	INTC (Software)
15	Software settable flag 15	INTC (Software)
16 - 31	Reserved	
32	Platform software watchdog timeout	Software watchdog timer
33 - 35	Reserved	
36	Platform periodic timer 0_0 (STM)	STM
37	Platform periodic timer 0_1 (STM)	STM
38	Platform periodic timer 0_2 (STM)	STM
39	Platform periodic timer 0_3 (STM)	STM
40 - 51	Reserved	
52	eDMA Combined Error	eDMA
53	eDMA Channel 0	eDMA
54	eDMA Channel 1	eDMA
55	eDMA Channel 2	eDMA
56	eDMA Channel 3	eDMA
57	eDMA Channel 4	eDMA
58	eDMA Channel 5	eDMA
59	eDMA Channel 6	eDMA

Table continues on the next page...

Table 7-16. Interrupt vector table (continued)

Vector	Source signal	Source module
60	eDMA Channel 7	eDMA
61	eDMA Channel 8	eDMA
62	eDMA Channel 9	eDMA
63	eDMA Channel 10	eDMA
64	eDMA Channel 11	eDMA
65	eDMA Channel 12	eDMA
66	eDMA Channel 13	eDMA
67	eDMA Channel 14	eDMA
68	eDMA Channel 15	eDMA
69	eDMA Channel 16	eDMA
70	eDMA Channel 17	eDMA
71	eDMA Channel 18	eDMA
72	eDMA Channel 19	eDMA
73	eDMA Channel 20	eDMA
74	eDMA Channel 21	eDMA
75	eDMA Channel 22	eDMA
76	eDMA Channel 23	eDMA
77	eDMA Channel 24	eDMA
78	eDMA Channel 25	eDMA
79	eDMA Channel 26	eDMA
80	eDMA Channel 27	eDMA
81	eDMA Channel 28	eDMA
82	eDMA Channel 29	eDMA
83	eDMA Channel 30	eDMA
84	eDMA Channel 31	eDMA
85 - 184	Reserved	
185	Flash controller Prog/Erase/Suspend IRQ_0	Platform flash controller
186 - 215	Reserved	
216	Timer IRQ Ethernet 0	ENET_0
217	TX IRQ Ethernet 0	ENET_0
218	RX IRQ Ethernet 0	ENET_0
219	Error/Generic IRQ1 Ethernet 0	ENET_0
220 - 223	Reserved	
Section B : off-platform common		
224	Reserved for Real Time Counter (RTC)	
225	Reserved for Autonomous Periodic Interrupt (API)	
226	Periodic Interrupt Timer (PIT_0) channel 0	PIT_0
227	Periodic Interrupt Timer (PIT_0) channel 1	PIT_0
228	Periodic Interrupt Timer (PIT_0) channel 2	PIT_0

Table continues on the next page...

Table 7-16. Interrupt vector table (continued)

Vector	Source signal	Source module
229	Periodic Interrupt Timer (PIT_0) channel 3	PIT_0
230 - 241	Reserved	
242	XOSC counter expired	External oscillator
243	SIUL2 External Interrupt_0	SIUL2
244	SIUL2 External Interrupt_1	SIUL2
245	SIUL2 External Interrupt_2	SIUL2
246	SIUL2 External Interrupt_3	SIUL2
247 - 250	Reserved	
251	Safe Mode Interrupt	MC_ME
252	Mode Transition Interrupt	MC_ME
253	Invalid Mode Interrupt	MC_ME
254	Invalid Mode Config	MC_ME
255	Functional and destructive reset alternate event interrupt	MC_RGM
256 - 258	Reserved	
259	DSPI_SR[TFUF] | DSPI_SR[RFOF]	DSPI0
260	DSPI_SR[EOQF]	DSPI0
261	DSPI_SR[TFFF]	DSPI0
262	DSPI_SR[TCF]	DSPI0
263	DSPI_SR[RFDF]	DSPI0
264 - 267	Reserved	
268	DSPI_SR[TFUF] | DSPI_SR[RFOF]	DSPI1
269	DSPI_SR[EOQF]	DSPI1
270	DSPI_SR[TFFF]	DSPI1
271	DSPI_SR[TCF]	DSPI1
272	DSPI_SR[RFDF]	DSPI1
273 - 276	Reserved	
277	DSPI_SR[TFUF] | DSPI_SR[RFOF]	DSPI2
278	DSPI_SR[EOQF]	DSPI2
279	DSPI_SR[TFFF]	DSPI2
280	DSPI_SR[TCF]	DSPI2
281	DSPI_SR[RFDF]	DSPI2
282 - 285	Reserved	
286	DSPI_SR[TFUF] | DSPI_SR[RFOF]	DSPI3
287	DSPI_SR[EOQF]	DSPI3
288	DSPI_SR[TFFF]	DSPI3
289	DSPI_SR[TCF]	DSPI3
290	DSPI_SR[RFDF]	DSPI3
291 - 375	Reserved	
376	LINFlex_RXI	LINFlex_0

Table continues on the next page...

Table 7-16. Interrupt vector table (continued)

Vector	Source signal	Source module
377	LINFlex_TXI	LINFlex_0
378	LINFlex_ERR	LINFlex_0
379	Reserved	
380	LINFlex_RXI	LINFlex_1
381	LINFlex_TXI	LINFlex_1
382	LINFlex_ERR	LINFlex_1
383 - 452	Reserved	
453	LRNEIF | DRNEIF	FlexRay_0
454	LRCEIF | DRCEIF	FlexRay_0
455	FNEAIF	FlexRay_0
456	FNEBIF	FlexRay_0
457	WUPIF	FlexRay_0
458	PRIF	FlexRay_0
459	CHIF	FlexRay_0
460	TBIF	FlexRay_0
461	RBIF	FlexRay_0
462	MIF	FlexRay_0
463 - 476	Reserved	
477	Power Monitor Unit	PMU
478	Power management Unit (temp sensor)	PMU
479 - 487	Reserved	
488	Alarm Interrupt (ALRM)	FCCU
489	Configuration Time-out (CFG_TO)	FCCU
490	Reserved for FCCU	FCCU
491	Reserved for FCCU	FCCU
492 - 495	Reserved	FCCU
Section C: off-platform device-specific modules		
496	ADC_EOC	ADC_0
497	ADC_ER	ADC_0
498	ADC_WD	ADC_0
499	Reserved	ADC_0
500	ADC_EOC	ADC_1
501	ADC_ER	ADC_1
502	ADC_WD	ADC_1
503	Reserved	ADC_1
504	ADC_EOC	ADC_2
505	ADC_ER	ADC_2
506	ADC_WD	ADC_2
507	Reserved	ADC_2

Table continues on the next page...

Table 7-16. Interrupt vector table (continued)

Vector	Source signal	Source module
508	ADC_EOC	ADC_3
509	ADC_ER	ADC_3
510	ADC_WD	ADC_3
511 - 519	Reserved	
520	FLEXCAN_ESR[ERR_INT]	FlexCAN_0
521	FLEXCAN_ESR_BOFF | FLEXCAN_Transmit_Warning | FLEXCAN_Receive_Warning	FlexCAN_0
522	FLEXCAN_BUF_00_03	FlexCAN_0
523	FLEXCAN_BUF_04_07	FlexCAN_0
524	FLEXCAN_BUF_08_11	FlexCAN_0
525	FLEXCAN_BUF_12_15	FlexCAN_0
526	FLEXCAN_BUF_16_31	FlexCAN_0
527	FLEXCAN_BUF_32_39	FlexCAN_0
528	FLEXCAN_BUF_40_47	FlexCAN_0
529	FLEXCAN_BUF_48_55	FlexCAN_0
530	FLEXCAN_BUF_56_63	FlexCAN_0
531 - 532	Reserved	FlexCAN_0
533	FLEXCAN_ESR[ERR_INT]	FlexCAN_1
534	FLEXCAN_ESR_BOFF | FLEXCAN_Transmit_Warning | FLEXCAN_Receive_Warning	FlexCAN_1
535	FLEXCAN_BUF_00_03	FlexCAN_1
536	FLEXCAN_BUF_04_07	FlexCAN_1
537	FLEXCAN_BUF_08_11	FlexCAN_1
538	FLEXCAN_BUF_12_15	FlexCAN_1
539	FLEXCAN_BUF_16_31	FlexCAN_1
540	FLEXCAN_BUF_32_39	FlexCAN_1
541	FLEXCAN_BUF_40_47	FlexCAN_1
542	FLEXCAN_BUF_48_55	FlexCAN_1
543	FLEXCAN_BUF_56_63	FlexCAN_1
544 - 545	Reserved	FlexCAN_1
546	FLEXCAN_ESR[ERR_INT]	FlexCAN_2
547	FLEXCAN_ESR_BOFF | FLEXCAN_Transmit_Warning | FLEXCAN_Receive_Warning	FlexCAN_2
548	FLEXCAN_BUF_00_03	FlexCAN_2
549	FLEXCAN_BUF_04_07	FlexCAN_2
550	FLEXCAN_BUF_08_11	FlexCAN_2
551	FLEXCAN_BUF_12_15	FlexCAN_2
552	FLEXCAN_BUF_16_31	FlexCAN_2
553	FLEXCAN_BUF_32_39	FlexCAN_2
554	FLEXCAN_BUF_40_47	FlexCAN_2

Table continues on the next page...

Table 7-16. Interrupt vector table (continued)

Vector	Source signal	Source module
555	FLEXCAN_BUF_48_55	FlexCAN_2
556	FLEXCAN_BUF_56_63	FlexCAN_2
557 - 569	Reserved	
570	Valid Fast Message Received on Channel 0	SENT_0
571	Valid Slow Message Received on Channel 0	SENT_0
572	Receive Error Interrupt Channel 0	SENT_0
573	Valid Fast Message Received on Channel 1	SENT_0
574	Valid Slow Message Received on Channel 1	SENT_0
575	Receive Error Interrupt Channel 1	SENT_0
576 - 581	Reserved	
582	Valid Fast Message Received on Channel 0	SENT_1
583	Valid Slow Message Received on Channel 0	SENT_1
584	Receive Error Interrupt Channel 0	SENT_1
585	Valid Fast Message Received on Channel 1	SENT_1
586	Valid Slow Message Received on Channel 1	SENT_1
587	Receive Error Interrupt Channel 1	SENT_1
588 - 593	Reserved	
594	Read interrupt channel 1	SIPI
595	Read interrupt channel 2	SIPI
596	Read interrupt channel 3	SIPI
597	Read interrupt channel 4	SIPI
598 - 601	Reserved	
602	Error 1	SIPI
603	Error 2	SIPI
604	Trigger command	SIPI
605	Tx interrupt	LFAST0
606	Tx exception	LFAST0
607	Rx interrupt	LFAST0
608	Rx exception	LFAST0
609	Rx ICLC	LFAST0
610	Reserved	LFAST0
611	TC0IR	eTimer_0
612	TC1IR	eTimer_0
613	TC2IR	eTimer_0
614	TC3IR	eTimer_0
615	TC4IR	eTimer_0
616	TC5IR	eTimer_0
617 - 618	Not used	Reserved eTimer_0
619	WTIF	eTimer_0

Table continues on the next page...

Table 7-16. Interrupt vector table (continued)

Vector	Source signal	Source module
620	Not used	Reserved eTimer_0
621	RCF	eTimer_0
622	TC0IR	eTimer_1
623	TC1IR	eTimer_1
624	TC2IR	eTimer_1
625	TC3IR	eTimer_1
626	TC4IR	eTimer_1
627	TC5IR	eTimer_1
628 - 631	Not used	Reserved eTimer_1
632	RCF	eTimer_1
633	TC0IR	eTimer_2
634	TC1IR	eTimer_2
635	TC2IR	eTimer_2
636	TC3IR	eTimer_2
637	TC4IR	eTimer_2
638	TC5IR	eTimer_2
639 - 642	Not used	Reserved eTimer_2
643	RCF	eTimer_2
644 - 649	Not used	eTimer_3
650 - 654	Not used	Reserved eTimer_3
655	RF0	FlexPWM_0
656	COF0	FlexPWM_0
657	CAF0	FlexPWM_0
658	RF1	FlexPWM_0
659	COF1	FlexPWM_0
660	CAF1	FlexPWM_0
661	RF2	FlexPWM_0
662	COF2	FlexPWM_0
663	CAF2	FlexPWM_0
664	RF3	FlexPWM_0
665	COF3	FlexPWM_0
666	CAF3	FlexPWM_0
667	FFLAG	FlexPWM_0
668	REF	FlexPWM_0
669	Reserved	FlexPWM_0
670	RF0	FlexPWM_1
671	COF0	FlexPWM_1
672	CAF0	FlexPWM_1
673	RF1	FlexPWM_1

Table continues on the next page...

Table 7-16. Interrupt vector table (continued)

Vector	Source signal	Source module
674	COF1	FlexPWM_1
675	CAF1	FlexPWM_1
676	RF2	FlexPWM_1
677	COF2	FlexPWM_1
678	CAF2	FlexPWM_1
679	RF3	FlexPWM_1
680	COF3	FlexPWM_1
681	CAF3	FlexPWM_1
682	FFLAG	FlexPWM_1
683	REF	FlexPWM_1
684	Reserved	FlexPWM_1
685 - 699	Reserved	FlexPWM_2
700	MRS_I	CTU_0
701	T0_I	CTU_0
702	T1_I	CTU_0
703	T2_I	CTU_0
704	T3_I	CTU_0
705	T4_I	CTU_0
706	T5_I	CTU_0
707	T6_I	CTU_0
708	T7_I	CTU_0
709	FIFO0_I	CTU_0
710	FIFO1_I	CTU_0
711	FIFO2_I	CTU_0
712	FIFO3_I	CTU_0
713	ADC_I	CTU_0
714	ERR_I	CTU_0
715	Reserved	CTU_0
716	MRS_I	CTU_1
717	T0_I	CTU_1
718	T1_I	CTU_1
719	T2_I	CTU_1
720	T3_I	CTU_1
721	T4_I	CTU_1
722	T5_I	CTU_1
723	T6_I	CTU_1
724	T7_I	CTU_1
725	FIFO0_I	CTU_1
726	FIFO1_I	CTU_1

Table continues on the next page...

Table 7-16. Interrupt vector table (continued)

Vector	Source signal	Source module
727	FIFO2_I	CTU_1
728	FIFO3_I	CTU_1
729	ADC_I	CTU_1
730	ERR_I	CTU_1
731	Reserved	CTU_1
732	sgen_error	SGEN
733	Reserved	SGEN
734-1023	Reserved	

7.4.6.3 INTC software vector mode and hardware vector mode

This information supplements and clarifies the information in these other locations:

- Interrupt Vector Prefix Registers (IVPR)
- Software vector mode
- Hardware vector mode

7.4.6.3.1 Vector table alignment

On this chip, the vector table must be aligned to 64 KB.

7.4.6.3.2 INTC software vector mode

In software vector mode, software—that is, the interrupt exception handler—reads an INTC register to obtain the vector associated with the interrupt request to the processor. The INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0) contains a 32-bit address composed of a vector table base address (VTBA) plus an offset that is the interrupt vector (INTVEC). The address is used to branch to the corresponding routine for that peripheral or software interrupt source.

The following figure shows a typical program flow for software vector mode.

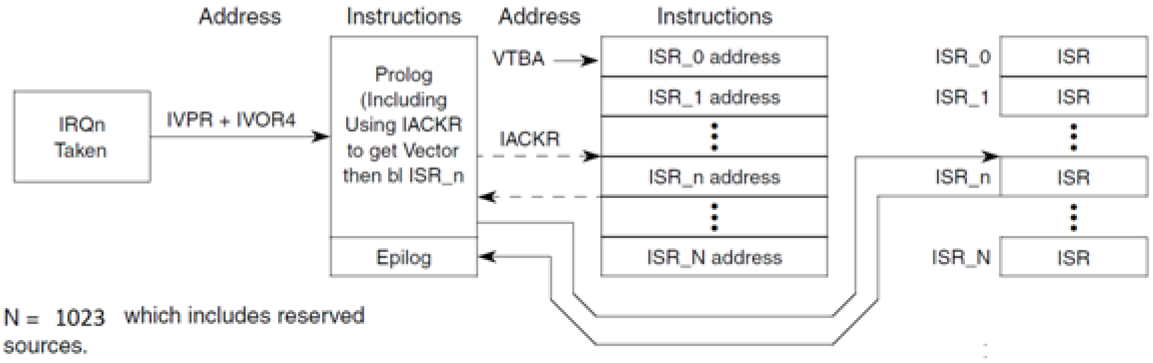


Figure 7-2. Software vector mode: program flow

In software vector mode, hardware calculates a common interrupt exception handler address as shown in the following figure. The upper part of the Interrupt Vector Prefix Register (IVPR) is added to the Interrupt Vector Offset Register 4 (IVOR4) offset.

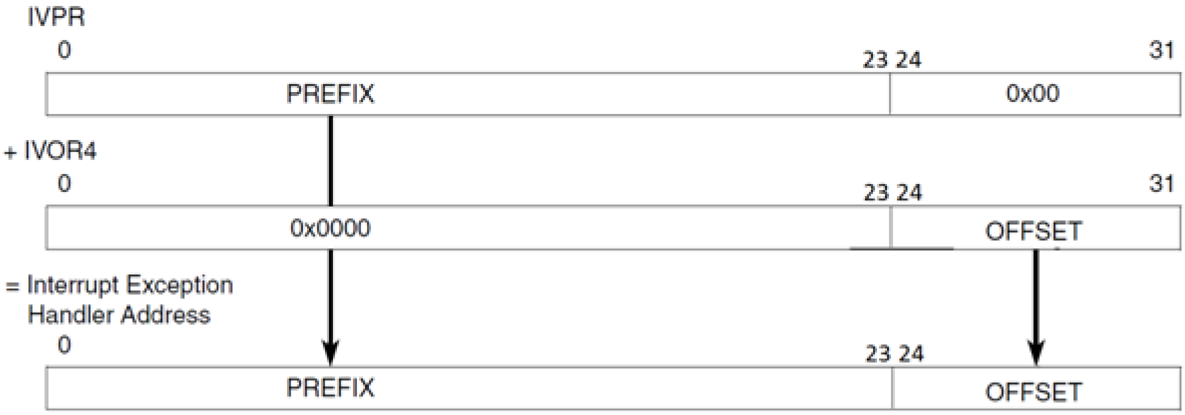


Figure 7-3. Software vector mode: interrupt exception handler address calculation

IVPR should be aligned to 256 bytes in software vector mode.

7.4.6.3.3 INTC hardware vector mode

In hardware vector mode, no Interrupt Vector Offset Register (IVOR) is used.

The following figure shows a typical program flow for hardware vector mode.

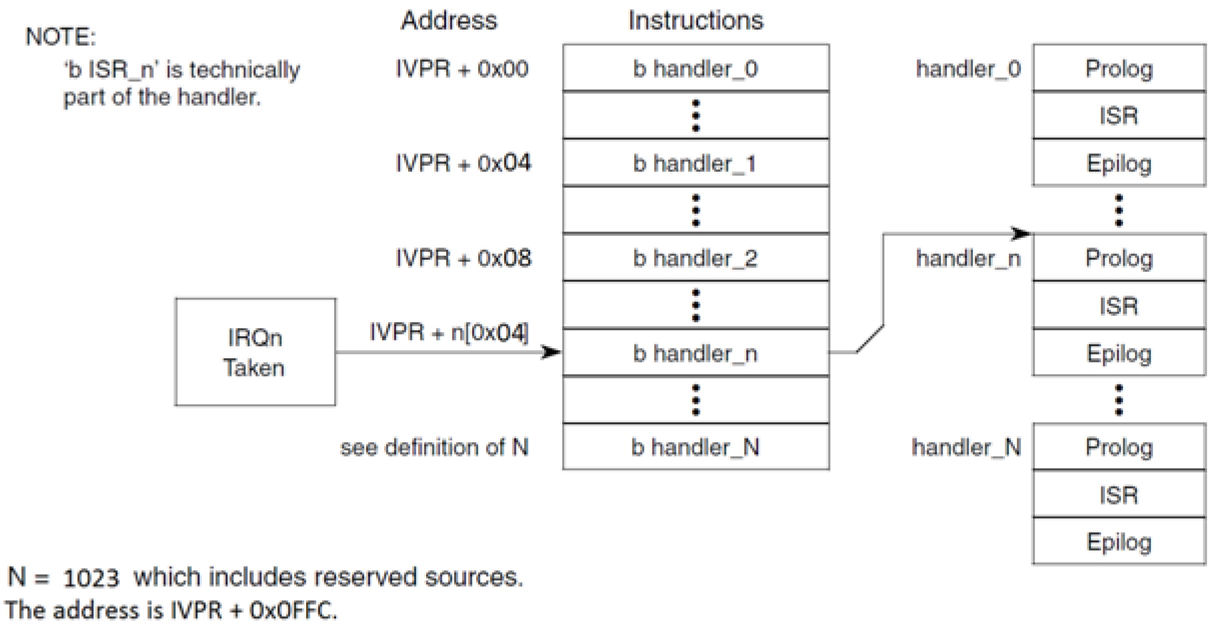


Figure 7-4. Hardware vector mode: program flow

In hardware vector mode, hardware calculates the interrupt exception handler address as shown in the following figure.

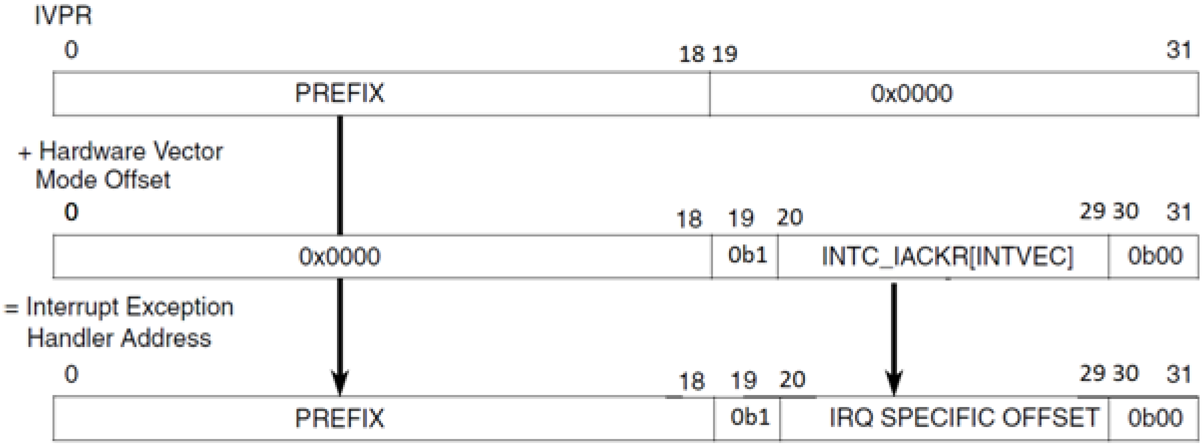


Figure 7-5. Hardware vector mode: interrupt exception handler address calculation

IVPR should be aligned to 8 KB in hardware vector mode.

7.4.7 DMA Controller configuration

The peripheral DMA requests are serviced by a DMA controller. A static multiplexing scheme allows the support of more peripheral requests than available DMA channels.

The device implements two 32-channel DMA controllers: DMA_0 and DMA_1. They are located in different peripheral lakes at the same location in the memory map, for redundancy.

DMA_1 is implemented in delayed lockstep using the same methodology as for the delayed lockstep of the cores:

- Only one DMA RAM array, which is accessed by DMA_0 only for read and write accesses
- DMA_1 is not visible to software. It cannot explicitly be accessed.
- DMA_1 operates with a delay of 2 cycles compared to DMA_0.
- The outputs of DMA_0 and DMA_1 are compared by RCCUs.
- The CORE_1 and DMA_1 are placed into the same lake.

For a comprehensive description of the DMA Controller, see its dedicated chapter.

7.4.8 DMAMUX configuration

This device contains two DMAMUX modules. DMAMUX_0 connects directly to DMA channels 0-15. DMAMUX_1 connects directly to DMA channels 16-31.

The DMAMUX does not generate any interrupt signals.

7.4.8.1 DMAMUX trigger inputs

The following table defines the signal sources for the trigger support of the first #TRG channels of the DMAMUX module. The table is valid for both instantiations of the module, DMAMUX_0 and DMAMUX_1.

Table 7-17. DMAMUX trigger sources

Source module	Source signal	DMAMUX channel trigger #
PIT	Trigger channel 0	0
PIT	Trigger channel 1	1
PIT	Trigger channel 2	2
PIT	Trigger channel 3	3

7.4.8.2 DMA request source slot mapping

The following tables define the mapping of the DMAMUX source slots to the interrupt request sources on the device.

Together, DMAMUX_0 and DMAMUX_1 support the total number of source slots documented in the dedicated DMAMUX chapter.

Table 7-18. DMAMUX_0 source slot mapping

DMAMUX source slot #	Source module	Source resource
1	DSPI_2	DSPI_TFFF
2	DSPI_2	DSPI_RFDF
3	DSPI_3	DSPI_TFFF
4	DSPI_3	DSPI_RFDF
5	CTU_0	CTU
6	CTU_0	FIFO1
7	CTU_0	FIFO2
8	CTU_0	FIFO3
9	CTU_0	FIFO4
10	FlexPWM_0	comp_val
11	FlexPWM_0	capt
12	eTimer_0	DREQ 0
13	eTimer_0	DREQ 1
14	eTimer_0	DREQ 2
15	eTimer_0	DREQ 3
16	eTimer_2	DREQ 0
17	eTimer_2	DREQ 1
18	ADC_0	DMA
19	ADC_2	DMA
20	LINFlex_0	Transmit
21	LINFlex_0	Receive
22	SENT_1	Fast message
23	SENT_1	Slow message
24	Always requester	—
25	Always requester	—
26	Always requester	—
27	Always requester	—
28	Always requester	—
29	Always requester	—

Table 7-19. DMAMUX_1 source slot mapping

DMAMUX source slot #	Source module	Source resource
1	DSPI_0	DSPI_TFFF
2	DSPI_0	DSPI_RFDF
3	DSPI_1	DSPI_TFFF
4	DSPI_1	DSPI_RFDF

Table continues on the next page...

Table 7-19. DMAMUX_1 source slot mapping (continued)

DMAMUX source slot #	Source module	Source resource
5	CTU_1	CTU
6	CTU_1	FIFO1
7	CTU_1	FIFO2
8	CTU_1	FIFO3
9	CTU_1	FIFO4
10	eTimer_1	DREQ 0
11	eTimer_1	DREQ 1
12	ADC_1	DMA
13	ADC_3	DMA
14	LINFlex_1	Transmit
15	LINFlex_1	Receive
16	FlexPWM_1	comp_val
17	FlexPWM_1	capt
18	SIPI	Channel 0
19	SIPI	Channel 1
20	SIPI	Channel 2
21	SIPI	Channel 3
22	SENT_0	Fast message
23	SENT_0	Slow message
24	SIUL2	Req 0
25	SIUL2	Req 1
26	SIUL2	Req 2
27	SIUL2	Req 3
28	Always requester	—
29	Always requester	—
30	Always requester	—
31	Always requester	—
32	Always requester	—
33	Always requester	—

7.4.9 Error Injection Module (EIM) configuration

The EIM provides error injection for the DMA memory Error Correcting Code (ECC).

7.5 Clocking

See the Clocking chapter for a description of the architecture for system level clocks, including clock generation, clock sources, and peripheral clocks. The chapter also provides configuration details for the Clock Monitor Unit (CMU).

7.6 Memories and memory interfaces

See the chapters shown in the following table for detailed information about memories and memory interfaces.

Table 7-20. Reference links to related information

Topic	Reference
Embedded memories	Embedded memories
System memory map	Memory Map
Platform Flash Memory Controller	Flash Memory Controller
Platform RAM Controller	RAM Controller (PRAM)
Embedded flash memory	Embedded Flash Memory (c55fmc)

7.6.1 RAM controller (PRAMC) configuration

7.6.1.1 PRAMC optional read wait states

The following table shows programming values for the Pn_BO_DIS and FT_DIS fields of the PRAMC's PRCR1 register and the resulting access timing for a WRAP4 burst. For additional information, see Optional read wait-state.

Table 7-21. Read wait states for WRAP4 burst

P0_BO_DIS or P1_BO_DIS	FT_DIS	WRAP4 timing
0	0	2/3-2-2-2
0	1	3/4-2-2-2
1	0	2/3-2-2-2
1	1	3/4-4-4-4

Programming additional, optional wait states is not required except as needed for debug.

7.6.2 Flash Memory Controller (PFLASH) configuration

The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the crossbar switch. The configuration and control functions are accessed via control registers, which are read/write accessible only in supervisor mode.

Three of the registers, PFCR1, PFCR2, and PFAPR, control the interaction of master modules with the flash array by enabling/disabling prefetch or by controlling access to the flash array on a per-master basis.

Use of the fields in these registers requires knowledge of the number assigned to each master. For example, the PFAPR[M2AP] field controls flash array access by master 2, the PFAPR[M3AP] field controls flash array access by master 3, and so on. See Logical master IDs for master assignments.

NOTE

ECC events encountered when the flash is in UTest mode are not subsequent to an externally-requested read operation. Margin Read and Array Integrity checks are strictly internal to the C55FMC flash and, as stated in the flash chapter, the results are combined by a MISR into a signature. ECC events during UTest activity are not reported externally to the flash.

See the PFCR3 register details in the flash memory controller chapter for more details on configuring the flash controller.

7.6.2.1 PFLASH interface to SRAM as overlay RAM

The PFLASH serves as the interface to the on-chip SRAM, which can be used as overlay RAM.

7.6.3 Embedded Flash Memory (c55fmc) configuration

7.6.3.1 C55FMC register reset values

The following table lists C55FMC register reset values that are specific to this chip. For the reset values of all other C55FMC registers, see Memory map and register definition.

Table 7-22. Chip-specific C55FMC register reset values

Register	Reset value
Extended Module Configuration Register (C55FMC_MCRE)	0460_0802h[1]
Over-Program Protection 0 register (C55FMC_OPP0) Over-Program Protection 1 register (C55FMC_OPP1) Over-Program Protection 2 register (C55FMC_OPP2) Over-Program Protection 3 register (C55FMC_OPP3)	See Flash OTP Control.

[1] This value applies for MPC5744P, MPC5743P, MPC5742P, and MPC5741P.

7.6.3.2 C55FMC_LOCK*n* register bit mapping

The C55FMC_LOCK*n* registers provide a means to protect flash memory blocks from being modified. Each implemented bit in a C55FMC_LOCK*n* register maps to a specific block.

7.6.3.2.1 C55FMC_LOCK0 register bit mapping

The flash memory blocks mapped to the Lock 0 register (C55FMC_LOCK0) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	TSLOCK	Reserved	LOWLOCK[13:0]														MIDLOCK[15:0]															

Table continues on the next page...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Flash Memory Block Name	UTest NVM Block Space 16 KB	—	—	—	—	—	—	—	—	—	—	—	16 KB Block 3 - Partition 1	16 KB Block 2 - Partition 1	16 KB EEPROM Data Block 1 - Partition 0	16 KB EEPROM Data Block 0 - Partition 0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	32 KB EEPROM Data Block 1 - Partition 3	32 KB EEPROM Data Block 0 - Partition 2

7.6.3.2.2 C55FMC_LOCK1 register bit mapping

The flash memory blocks mapped to the Lock 1 register (C55FMC_LOCK1) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved																HIGHLOCK[15:0]															
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	64 KB Block 5 - Partition 5	64 KB Block 4 - Partition 5	64 KB Block 3 - Partition 5	64 KB Block 2 - Partition 4	64 KB Block 1 - Partition 4	64 KB Block 0 - Partition 4

7.6.3.2.3 C55FMC_LOCK2 register bit mapping

The flash memory blocks mapped to the Lock 2 register (C55FMC_LOCK2) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	A256KLOCK[31:0]																															
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	256 KB Block 7 - Partition 7	256 KB Block 6 - Partition 7[1]	256 KB Block 5 - Partition 7	256 KB Block 4 - Partition 7[2]	256 KB Block 3 - Partition 6	256 KB Block 2 - Partition 6[3]	256 KB Block 1 - Partition 6	256 KB Block 0 - Partition 6

1. This block is reserved on MPC5741P, MPC5742P, and MPC5743P.
2. This block is reserved on MPC5741P and MPC5742P.
3. This block is reserved on MPC5741P.

7.6.3.2.4 C55FMC_LOCK3 register bit mapping

The flash memory blocks mapped to the Lock 3 register (C55FMC_LOCK3) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved																A256KLOCK[15:0]															

Table continues on the next page...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

7.6.3.3 C55FMC_SEL*n* register bit mapping

The C55FMC_SEL*n* registers provide a means to select flash memory blocks to be erased. Each implemented bit in a C55FMC_SEL*n* register maps to a specific memory block.

7.6.3.3.1 C55FMC_SEL0 register bit mapping

The flash memory blocks mapped to the Select 0 register (C55FMC_SEL0) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved		LOWSEL[13:0]														MIDSEL[15:0]															

Table continues on the next page...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Flash Memory Block Name	–	–	–	–	–	–	–	–	–	–	–	–	16 KB Block 3 - Partition 1	16 KB Block 2 - Partition 1	16 KB EEPROM Data Block 1 - Partition 0	16 KB EEPROM Data Block 0 - Partition 0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	32 KB EEPROM Data Block 1 - Partition 3	32 KB EEPROM Data Block 0 - Partition 2

7.6.3.3.2 C55FMC_SEL1 register bit mapping

The flash memory blocks mapped to the Select 1 register (C55FMC_SEL1) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved																HIGHSEL[15:0]															
Flash Memory Block Name	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	64 KB Block 5 - Partition 5	64 KB Block 4 - Partition 5	64 KB Block 3 - Partition 5	64 KB Block 2 - Partition 4	64 KB Block 1 - Partition 4	64 KB Block 0 - Partition 4

7.6.3.3.3 C55FMC_SEL2 register bit mapping

The flash memory blocks mapped to the Select 2 register (C55FMC_SEL2) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	A256KSEL[31:0]																															
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	256 KB Block 7 - Partition 7	256 KB Block 6 - Partition 7 [1]	256 KB Block 5 - Partition 7	256 KB Block 4 - Partition 7 [2]	256 KB Block 3 - Partition 6	256 KB Block 2 - Partition 6 [3]	256 KB Block 1 - Partition 6	256 KB Block 0 - Partition 6

1. This block is reserved on MPC5741P, MPC5742P, and MPC5743P.
2. This block is reserved on MPC5741P and MPC5742P.
3. This block is reserved on MPC5741P.

7.6.3.3.4 C55FMC_SEL3 register bit mapping

The flash memory blocks mapped to the Select 3 register (C55FMC_SEL3) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved																A256KSEL[15:0]															

Table continues on the next page...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

7.6.3.4 C55FMC_OPP*n* register bit mapping

The C55FMC_OPP*n* registers provide a means to protect blocks from being over-programmed. Each implemented bit in a C55FMC_OPP*n* register maps to a specific memory block.

7.6.3.4.1 C55FMC_OPP0 register bit mapping

The flash memory blocks mapped to the Over-Program Protection 0 register (C55FMC_OPP0) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved	Reserved	LOWOPP[13:0]														MIDOPP[15:0]															

Table continues on the next page...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	16 KB Block 3 - Partition 1	16 KB Block 2 - Partition 1	16 KB EEPROM Data Block 1 - Partition 0	16 KB EEPROM Data Block 0 - Partition 0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	32 KB EEPROM Data Block 1 - Partition 3	32 KB EEPROM Data Block 0 - Partition 2

7.6.3.4.2 C55FMC_OPP1 register bit mapping

The flash memory blocks mapped to the Over-Program Protection 1 register (C55FMC_OPP1) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved																HIGHOPP[15:0]															
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	64 KB Block 5 - Partition 5	64 KB Block 4 - Partition 5	64 KB Block 3 - Partition 5	64 KB Block 2 - Partition 4	64 KB Block 1 - Partition 4	64 KB Block 0 - Partition 4

7.6.3.4.3 C55FMC_OPP2 register bit mapping

The flash memory blocks mapped to the Over-Program Protection 2 register (C55FMC_OPP2) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	A256KOPP[31:0]																															
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	256 KB Block 7 - Partition 7	256 KB Block 6 - Partition 7 [1]	256 KB Block 5 - Partition 7	256 KB Block 4 - Partition 7 [2]	256 KB Block 3 - Partition 6	256 KB Block 2 - Partition 6 [3]	256 KB Block 1 - Partition 6	256 KB Block 0 - Partition 6

1. This block is reserved on MPC5741P, MPC5742P, and MPC5743P.
2. This block is reserved on MPC5741P and MPC5742P.
3. This block is reserved on MPC5741P.

7.6.3.4.4 C55FMC_OPP3 register bit mapping

The flash memory blocks mapped to the Over-Program Protection 3 register (C55FMC_OPP3) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved																A256KOPP[15:0]															

Table continues on the next page...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

7.6.4 Decorated Storage Memory Controller (DSMC) configuration

The DSMC supports five store and three load operations, including:

Bit field inserts
Compare-and-store
Bitwise AND, OR, and XOR operators
Simple memory load
Swap
Load-and-set-1(bit)

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

The module includes error checking logic that validates the decoration field, ensuring it is properly defined: all illegal commands are rejected and the transfer error is terminated. Additionally, the decorated storage memory controller only operates on aligned, single data transfers. Misalignment and/or bursts are not supported and, if attempted, are error terminated.

The DSMC is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

The device DSMC controllers are instantiated for SRAM, PBRIDGE_0, and PBRIDGE_1. The access size is forced to 64-bit for SRAM and 32-bit for the PBRIDGE modules.

7.7 Motor control configurations

This section provides information regarding various frequency combinations that can be used depending on the used case. The table shown below gives the valid CGM configurations for motor control subsystem.

Table 7-23. Valid CGM configurations for motor control subsystem

Clock domain	Config 1	Config 2	Config 3	Config 4
SYS_CLK	PLL1_200 MHz	PLL1_200 MHz	PLL1_200 MHz	PLL0_200 MHz
AC0_DC0 (ADC aux clk divider)	PLL0_160 MHz	PLL1_200/2 MHz = 100 MHz	PLL0_200/2 MHz = 100 MHz	PLL0_200/2 MHz = 100 MHz
AC0_DC2 (ADC aux clk divider)	PLL0_160/2 MHz = 80 MHz	PLL1_200/4 MHz = 50 MHz	PLL0_200/4 MHz = 50 MHz	PLL0_200/4 MHz = 50 MHz

Example used cases:

- For best ADC dynamic performance, it is recommended to run with single PLL supplying both system clock and motor control clocks (either PLL1 - Config 2 or PLL0 - Config 4).
- In case dual PLL configuration is required, run at a frequency where the clocks are harmonically related (Config - 3).
- Use Config - 1, i.e PLL1 with frequency modulation to improve EMC and PLL0 with frequency modulation for noise free usage of motor control peripherals and ADC or when Flexray with 80 MHz configuration is required.

7.8 Motor control modules

This section highlights the modules for motor control that are implemented on the device.

Table 7-24. Motor control modules

Module or subsystem	Device-specific information	Block details
Temperature sensor	Temperature sensor configuration	Temperature Sensor
ADC subsystem	ADC Configuration chapter	See individual modules
Successive Approximation Register Analog-to-Digital Converters (SAR ADCs)	ADC Configuration chapter	ADC Digital (ADCD)
eTimer	Enhanced Motor Control Timer (eTimer) configuration	Enhanced Motor Control Timer (eTimer)
FlexPWM	Motor Control Pulse Width Modulator Module (FlexPWM) configuration	Motor Control Pulse Width Modulator Module (FlexPWM)
CTU	Cross-Triggering Unit (CTU) configuration	Cross-Triggering Unit (CTU)

7.8.1 Temperature sensor configuration

This device contains two Temperature Sensor modules, TSENS_0 and TSENS_1, located in different peripheral lakes.

The junction temperature sensor generates an output voltage that is directly proportional to the internal junction temperature of the device. The information provided by this sensor can be used to implement some intelligent power control (such as to reduce frequency when temperature is too high).

The mapping of the two temperature sensors to the ADC channels is defined in the ADC Configuration details. The temperature sensor has no input signals. Out of bounds detection of the chip temperature is handled via:

- A/D conversion of the analog values of both temperature sensors and software evaluation and software-initiated action on the results.
- FCCU failure generation according to the defined low and high temperature points.

7.8.1.1 Temperature sensor registers

The Power Management Controller (PMC) module provides registers controls for the temperature sensor modules. See the chapter dedicated to the PMC.

7.8.1.2 Temperature sensor bandgap reference

The analog portion of the PMC provides the bandgap reference for the temperature sensor modules.

7.8.2 Analog to Digital Converter (ADC) configuration

See ADC Configuration.

7.8.3 Sine Wave Generator (SGEN) configuration

This chip implements one instance of the SGEN module, SGEN_0.

7.8.4 Enhanced Motor Control Timer (eTimer) configuration

This device contains three eTimer modules: eTimer_0, eTimer_1, and eTimer_2.

Each eTimer module comprises six 16-bit channels. All eTimer modules supports 5 Auxiliary Inputs. eTimer_0 supports the watchdog timer function.

All eTimers have DMA support. eTimer_0 has 4 DMA channels (channel 3,2,1,0). eTimer_1 and eTimer_2 each have 2 DMA channels (channel 1,0).

7.8.4.1 Required eTimer configuration before STOP mode entry

When ETIMER_CHn_CTRL3[STPEN] for an eTimer instance is 1: Before entering STOP mode, configure the eTimer to be stopped by programming MC_ME_PCTL137[LP_CFG] (eTimer_1), MC_ME_PCTL245[LP_CFG] (eTimer_2), or MC_ME_PCTL247[LP_CFG] (eTimer_0), as applicable, to select an MC_ME_LP_PCn configuration where MC_ME_LP_PCn[STOP0] is 0.

7.8.5 Motor Control Pulse Width Modulator (FlexPWM) configuration

This device contains two FlexPWM modules: FlexPWM_0 and FlexPWM_1.

The following table shows all signals connected to FlexPWM_0 and FlexPWM_1 input ports originating from other modules on the SoC for inter-module communication. See CTU inter-module connections for more details.

Table 7-25. FlexPWM_0 and FlexPWM_1 module-to-module input signals

Source (output ports)		Destination (input ports)	
Module name	**Port name**	**Module name**	**Port name**
eTimer_0	T1	FlexPWM_0	EXT_FORCE
eTimer_1	T1	FlexPWM_0	CLOCK
eTimer_2	T1	FlexPWM_1	EXT_FORCE
eTimer_2	T3	FlexPWM_1	CLOCK
FlexPWM_0	Master Reload (local_reload0)	FlexPWM_1	EXT_SYNC

7.8.6 Cross-Triggering Unit (CTU) configuration

This device contains two CTU modules: CTU_0 and CTU_1.

7.8.6.1 CTU inter-module connections

The following figure shows the inter-module connection of the CTUs for the Motor Control related peripherals for the device.

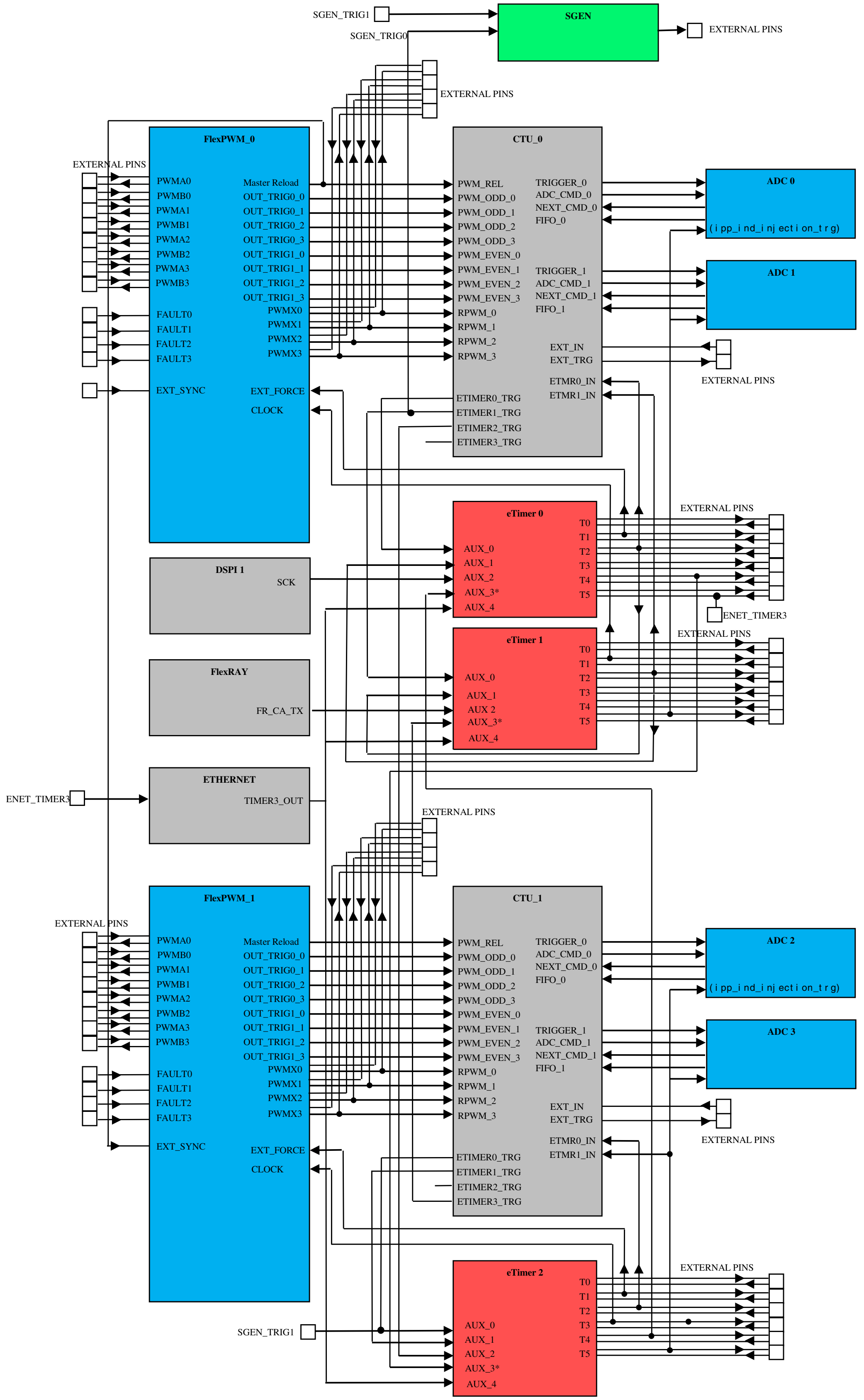


Figure 7-6. CTU inter-module connections

7.8.6.2 TGSISR input assignments for CTU_0 and CTU_1

The following table identifies the alignment among the following:

- The input number used in the I*n*RE and I*n*FE fields of the CTU's register TGSISR
- The generic name used in the CTU chapter for the CTU input with that input number
- For both CTU_0 and CTU_1, the specific source on this device for the CTU input with that input number

Table 7-26. Input assignments for CTU_0_TGSISR and CTU_1_TGSISR

Input	TGSISR fields	CTU generic input name	CTU_0_TGSISR source	CTU_1_TGSISR source
0	I0RE and I0FE	PWM_REL	FlexPWM_0 Master Reload	FlexPWM_1 Master Reload
1	I1RE and I1FE	PWM_ODD_0	FlexPWM_0 OUT_TRIG0_0	FlexPWM_1 OUT_TRIG0_0
2	I2RE and I2FE	PWM_ODD_1	FlexPWM_0 OUT_TRIG0_1	FlexPWM_1 OUT_TRIG0_1
3	I3RE and I3FE	PWM_ODD_2	FlexPWM_0 OUT_TRIG0_2	FlexPWM_1 OUT_TRIG0_2
4	I4RE and I4FE	PWM_ODD_3	FlexPWM_0 OUT_TRIG0_3	FlexPWM_1 OUT_TRIG0_3
5	I5RE and I5FE	PWM_EVEN_0	FlexPWM_0 OUT_TRIG1_0	FlexPWM_1 OUT_TRIG1_0
6	I6RE and I6FE	PWM_EVEN_1	FlexPWM_0 OUT_TRIG1_1	FlexPWM_1 OUT_TRIG1_1
7	I7RE and I7FE	PWM_EVEN_2	FlexPWM_0 OUT_TRIG1_2	FlexPWM_1 OUT_TRIG1_2
8	I8RE and I8FE	PWM_EVEN_3	FlexPWM_0 OUT_TRIG1_3	FlexPWM_1 OUT_TRIG1_3
9	I9RE and I9FE	RPWM_0	FlexPWM_0 PWMX0	FlexPWM_1 PWMX0
10	I10RE and I10FE	RPWM_1	FlexPWM_0 PWMX1	FlexPWM_1 PWMX1
11	I11RE and I11FE	RPWM_2	FlexPWM_0 PWMX2	FlexPWM_1 PWMX2
12	I12RE and I12FE	RPWM_3	FlexPWM_0 PWMX3	FlexPWM_1 PWMX3
13	I13RE and I13FE	ETIMER1_IN	eTimer_0 TO2	eTimer_2 TO2
14	I14RE and I14FE	ETIMER2_IN	eTimer_1 TO2	eTimer_2 TO5
15	I15RE and I15FE	EXT_IN	External pin	External pin

7.9 Timers

Three types of timers and a complex timer subsystem are implemented on the device.

Table 7-27. Timers

Timer	Device-specific information	Block details
System Timer Module (STM)	System Timer Module (STM) configuration	System Timer Module (STM)
Software Watchdog Timer (SWT)	Software Watchdog Timer (SWT) configuration	Software Watchdog Timer (SWT)
Periodic Interrupt Timer (PIT)	PIT configuration	Periodic Interrupt Timer (PIT)

Table continues on the next page...

Table 7-27. Timers (continued)

Timer	Device-specific information	Block details
Enhanced Motor Control Timer (eTimer)	Enhanced Motor Control Timer (eTimer) configuration	Enhanced Motor Control Timer (eTimer)

The master core has a dedicated System Timer Module (STM) and a dedicated Software Watchdog Timer (SWT).

7.9.1 System Timer Module (STM) configuration

This device contains one System Timer Module (STM).

The STM chapter states that the internal timer of the STM runs on the "system clock." On this chip, the STM's "system clock" is PBRIDGEx_CLK.

On this chip, STM_CR resets to 0000_0000h.

7.9.2 Software Watchdog Timer (SWT) configuration

This device contains one Software Watchdog Timer (SWT) module.

The internal timer of the SWT always runs on the IRCOSC clock.

The user must configure the outcomes of SWT time-out events by programming ITR in the SWT_CR as well as FCCU reactions to associated non-critical faults (NCFs):

- When SWT_CR[ITR] is 0, any SWT time-out generates a reset request that corresponds to NCF[14].
- When SWT_CR[ITR] is 1:
  - An initial time-out generates an interrupt request and triggers NCF[72].
  - A second consecutive time-out generates a reset request that corresponds to NCF[14].

CAUTION

To ensure that NCF[14] results in a reset, configure the FCCU’s reaction to NCF[14] to be a long functional reset or short functional reset.

7.9.3 Periodic Interrupt Timer (PIT) configuration

This device contains one Periodic Interrupt Timer (PIT) module.

The PIT contains four 32-bit timer channels. Four channels are used as trigger timer for DMAMUX.

NOTE

For valid PIT clock frequency for this device please refer to Figure 13-3 and Table: Maximum system level clock frequencies.

7.10 Communication interfaces

7.10.1 FlexCAN configuration

This device contains three Controller Area Network (CAN) modules: FlexCAN_0, FlexCAN_1, and FlexCAN_2. Each FlexCAN instantiation supports 64 Message Buffers.

Each FlexCAN module supports programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator.

Each FlexCAN module sources 11 interrupt vectors.

Each FlexCAN module has one external output CAN_Tx (CAN Transmit) and one external input CAN_Rx (CAN Receive).

7.10.1.1 FlexCAN interrupt flag ERRSR[CEIF]

The interrupt flag ERRSR[CEIF] does not have an assigned vector and does not generate an interrupt because the error it flags is corrected automatically. Nevertheless, ERRSR[CEIF] can be used as a status flag to report single-bit correctable memory errors.

7.10.2 LVDS Fast Asynchronous Transmission (LFAST) interface configuration

This device contains one LFAST interface.

The LFAST module is used to transfer data and control information between two devices (Master and Slave), and is capable of supporting either Dual mode (for example, configurable Master/Slave) or Slave-only mode functionality.

The serial link can run at up to 320 MBit/s using an LVDS physical layer. The following external signals are needed:

- One LVDS pair for Tx
- One LVDS pair for Rx
- Reference voltage pad
- Separate reference clock signal (not LVDS)

The LFAST Tx and Rx lines are connected to the pads via the SIUL2 and the MSCR registers. The Tx Data Port/Rx Data Port is connected to the SIPI module. The LFAST supports five interrupts.

The LFAST implementation includes a dedicated PLL for multiplying a low-speed clock source up to 320 MHz for the transmission and reception of data. The high-speed asynchronous nature of LFAST requires that both the local node and remote node operate from the same stable, low-jitter clock sources. This common low-speed clock is referred to as the Reference Clock.

7.10.3 Serial Peripheral Interface (SPI) configuration

This chip has four SPI module instances.

- None of the instances supports or uses these features:
  - Deserialization (DSI)
  - Global interrupt request line
- SPI_0 and SPI_1 each support 8 chip selects. SPI_2 and SPI_3 each support only 4 chip selects.

The terms SPI and DSPI are used interchangeably in this document.

7.10.4 FlexRay configuration

This device contains one dual channel FlexRay module with 64 configurable message buffers. The FlexRay module provides the following features:

- Protocol implementation compliant with FlexRay Communications System Protocol Specification, Version 2.1 Rev A
- Bus driver interface compliant with FlexRay Communications System Electrical Physical Layer Specification, Version 3.0
- Can be configured for single channel

- FlexRay Port A can be configured to be connected either to physical FlexRay channel A or physical FlexRay channel B
- Support for FlexRay bus data rates of 10 Mbit/s, 8 Mbit/s, 5 Mbit/s, and 2.5 Mbit/s

7.10.4.1 FlexRay interrupts

The FlexRay module provides seven individual interrupt sources and five combined interrupt sources. The following table describes all twelve interrupt sources.

- Eight interrupt sources are routed to the INTC.
- Two interrupt sources indicating correctable ECC errors of the local SRAM are combined (ORed) and then routed to the INTC. One of these two is also routed to the MEMU.
- Two interrupts indicating non-correctable ECC errors of the local SRAM are routed individually to the MEMU as well as combined (ORed) and routed to the INTC.

Table 7-28. FlexRay interrupts

Interrupt name	Register field	FlexRay module internal IRQ type	Connected to
Receive FIFO Channel A Not Empty Interrupt	FNEAIF	Individual	INTC
Receive FIFO Channel B Not Empty Interrupt	FNEBIF	Individual	INTC
Wakeup Interrupt	WUPIF	Individual	INTC
Protocol Status and Error Interrupt	PRIF	Combined	INTC
CHI Error Interrupt	CHIF	Combined	INTC
Transmit Message Buffer Interrupt	TBIF	Combined	INTC
Receive Message Buffer Interrupt	RBIF	Combined	INTC
Module Interrupt	MIF	Combined	INTC
LRAM Correctable ECC Error Interrupt	LRCEIF	Individual	INTC
DRAM Correctable ECC Error Interrupt	DRCEIF[1]	Individual	INTC, MEMU
LRAM Non-correctable ECC Error Interrupt	LRNEIF	Individual	INTC, MEMU
DRAM Non-correctable ECC Error Interrupt	DRNEIF[2]	Individual	

[1] These two individual interrupt sources are combined (logical OR) before being routed to the interrupt controller.

[2] These two individual interrupt sources are combined (logical OR) before being routed to the interrupt controller. They are routed individually to the MEMU.

7.10.4.2 FlexRay external signals

Table 7-29. FlexRay external signals

Signal name	Direction	Signal description
FR_A_TX	Output	FlexRay Channel A Transmit
FR_A_TX_EN	Output	FlexRay Channel A Transmit Enable: active low

Table continues on the next page...

Table 7-29. FlexRay external signals (continued)

Signal name	Direction	Signal description
FR_A_RX	Input	FlexRay Channel A Receive
FR_B_TX	Output	FlexRay Channel B Transmit
FR_B_TX_EN	Output	FlexRay Channel B Transmit Enable: active low
FR_B_RX	Input	FlexRay Channel B Receive
FR_DBG[3:0]	Output	Debug/Strobe Output (shared with GPIO)

7.10.5 SENT receiver (SRX) configuration

The Single Edge Nibble Transmission (SENT) module is designed as a receiver for transmitting sensor nodes using the SENT protocol (as documented by the SAE J2716 information report). SENT is a unidirectional communications scheme from a sensor (transmitting) device to a controller (receiving) device that does not include a coordination signal from the controller/receiving device.

This device contains two SENT modules. Each has two channels. The two modules are distributed to the two peripheral subsystems to allow redundancy for safety applications. The SENT has interrupts for slow and fast messages. There are two DMA requests per instantiated SENT module for the fast and slow messages. The SENT FIFO depth is 4.

The 5.0 V power supply required for a SENT interface is not provided by the MCU. This supply must be provided externally as shown in SAE J2716. As a result, the SENT has only one receive input pad per channel.

7.10.6 LINFlexD configuration

This device contains two LINFlex peripherals: LINFlex_0 and LINFlex_1. They are distributed between the two AIPS peripheral sub system for safety reasons.

The LINFlex module supports master and slave mode, contains one TX and one RX channel, and contains 16 identifier filtering in hardware.

Each LINFlex module can generate interrupts (RX, TX, Error). TX and RX have DMA channels.

The LINFlex baud rate clock is half the system clock. For example: When the system clock is 200 MHz, the LINFlex baud rate clock is 100 MHz.

7.10.7 Ethernet (ENET) configuration

The ENET module is connected to the eTimer modules. See the CTU inter-module connections diagram.

NOTE

> Before making any access to the Ethernet registers make sure the ENET module clock is enabled. If not, it is possible for the system to stall instead of generating a bus error.

The ENET module connects to external timer signals. See the Clocking chapter for details.

The ENET module's RMII_CLK can be output on Port Pin G[8]. Along with configuring the SIUL2_MSCR104 for selecting the ENET_0[RMII_CLK] function on Port Pin G[8], the user also needs to configure GCR[RMII_CLK_EN].

7.10.7.1 Generic Control Register (GCR)

	0xFFC0_C000																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																										RMII_CLK_EN	Reserved				
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Field	Description
0 - 25 Reserved	This field is reserved.
26 RMII_CLK_EN	Configuration bit that must be programmed along with SIUL2_MSCR[104] to enable RMII_CLK on pad[104] (Port Pin G[8]). 0 - RMII_CLK (Aux Clock 10) is not output on pad[104] (Port Pin G[8]) 1 - RMII_CLK (Aux Clock 10) is output on pad[104] (Port Pin G[8])
27 - 31 Reserved	This field is reserved.

7.11 Reset and boot modules

7.11.1 System Status and Configuration Module (SSCM) configuration

There is one SSCM instance in the device. The SSCM is part of the reset and boot subsystem.

NOTE

UART and SCI are used interchangeably in the document beyond this point.

7.11.1.1 SSCM_MEMCONFIG reset value

On the production-intent version of the chip, these MEMCONFIG fields have the indicated reset values:

- MREV: 2h
- JPIN: 2C5h

7.11.1.2 SSCM_UOPS register and soc_conf DCF client

On this device, two fields in the SSCM_UOPS register perform a specific function on the device. The other fields in the register are reserved.

The DCF client soc_conf provides the value of the SSCM_UOPS fields. For information about modifying soc_conf, see Device Configuration Format (DCF) Records.

The following table defines the two fields and shows their bit positions in both the register and the DCF client.

Table 7-30. SSCM_UOPS and soc_conf fields

Field	SSCM_UOPS bit position	soc_conf bit position	Default value	Function of default value
Checker Core Enable (CCE)	2	31	1	Enables checker core[1]
Watchdog Enable (WDE)	0	30	0	Enables watchdog automatically after device exits reset

[1] Turn off checker core for power saving or debug

7.11.1.3 UTEST_OFFSET location

On this device, the UTEST_OFFSET location at which the SSCM looks for UTEST DCF records is 0040_0200h.

7.11.1.4 SSCM_ERROR[RAE] exception not produced for some modules

Setting RAE in the SSCM Error Configuration Register (SSCM_ERROR) to 1 produces a Prefetch or Data Abort exception for illegal accesses to most off-platform peripherals.

However, for a subset of off-platform peripherals, illegal accesses to unimplemented spaces in the peripherals' memory maps do not produce a Prefetch or Data Abort exception. This subset includes the CMUs, SIPI, FCCU and CTU.

7.11.1.5 SSCM boot locations in flash memory

Table 5-4 identifies the boot locations in flash memory that contain a valid RCHW record.

7.12 Safety modules

7.12.1 CRC configuration

The value of the parameter CRC_CNTX_NUM, which defines the number of application contexts and therefore the number of instances of CRC registers, is 3.

7.12.2 Memory Error Management Unit (MEMU) configuration

The following diagram shows the data flow of the various Error Correction Code logic used for the Flash, System RAM and the Peripheral RAMs.

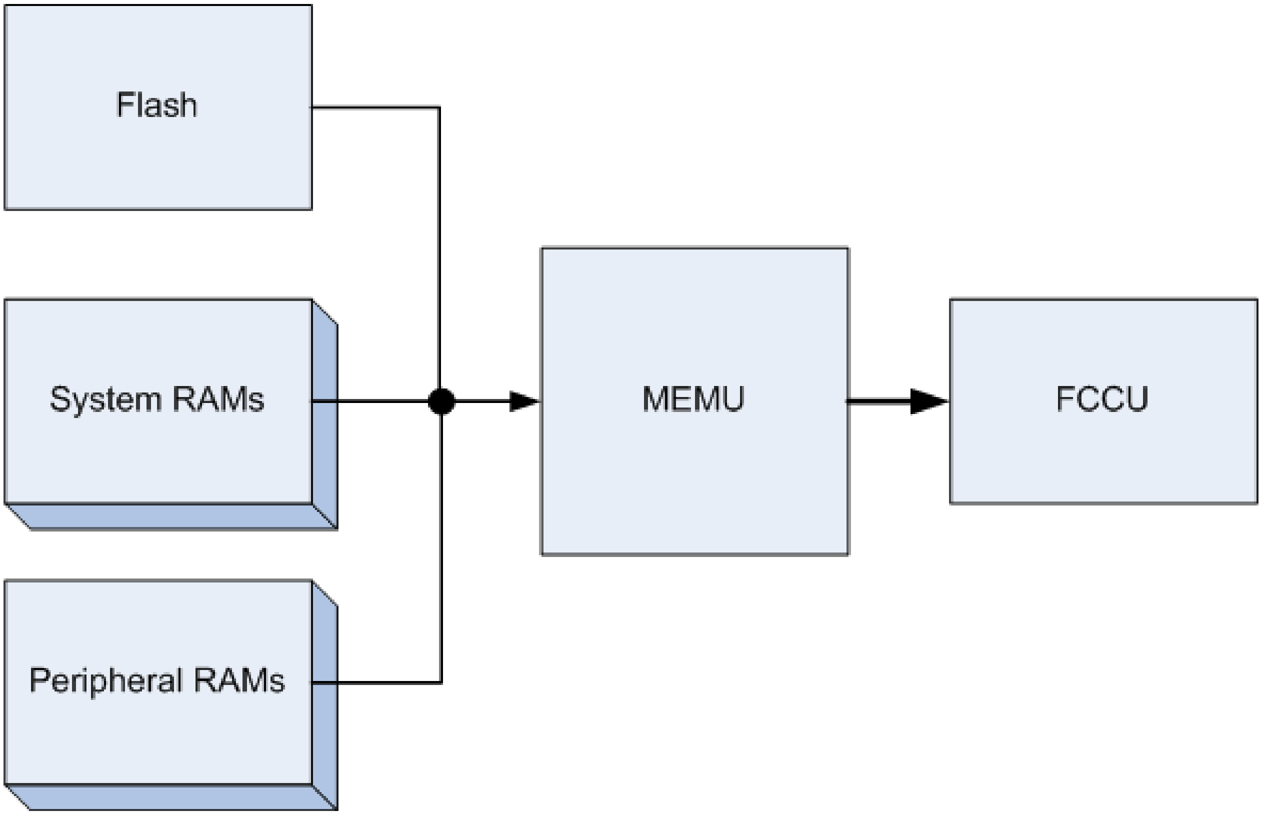


Figure 7-7. MEMU System Connections

7.12.2.1 Error sources for Concurrent Overflow (OFLWn) registers

The three categories of error reporting for the MEMU are:

- System RAM error reporting: Any RAM that is CPU accessible falls into this category.
- Peripheral RAM error reporting: Any RAM that is not CPU accessible falls into this category.
- Flash error reporting

In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLWn register details the corresponding error source to the respective field in the Concurrent Overflow registers.

Table 7-31. Corresponding error sources for OFLWn registers

Memory	Memory Classification	ECC: Single Error Correcting (SEC) or Double Error Detecting (DED)	OFLW register: <bit position>
Core 0			
Core D-MEM CPU0	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 0
Core I-Cache	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 1

Table continues on the next page...

Table 7-31. Corresponding error sources for OFLWn registers (continued)

Memory	Memory Classification	ECC: Single Error Correcting (SEC) or Double Error Detecting (DED)	OFLW register: <bit position>
Core D-Cache[1]	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 2
I-AHB E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 3
D-AHB E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 4
Platform			
PRAM E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 5
FlexRay E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 6
PBRIDGE0 E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 7
PBRIDGE1 E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 8
DMA E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 9
SIPI E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 10
D-TCM Backdoor E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 11
DSMC PRAM E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 12
DSMC PBRIDGE0 E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0: bit 13
DSMC PBRIDGE1 E2E ECC	System RAM	SEC/DED	MEMU_SYS_RAM_OFLW0 : bit 14
FlexRay			
FlexRay DRAM	Peripheral RAM	SEC/DED	MEMU_PERIPH_RAM_OFLW0: bit 0
FlexRay LRAM	Peripheral RAM	Error Detection Code (EDC) Only	MEMU_PERIPH_RAM_OFLW0: bit 1
FlexCAN			
FlexCAN 0	Peripheral RAM	SEC/DED	MEMU_PERIPH_RAM_OFLW0: bit 2
FlexCAN 1	Peripheral RAM	SEC/DED	MEMU_PERIPH_RAM_OFLW0: bit 3
FlexCAN 2	Peripheral RAM	SEC/DED	MEMU_PERIPH_RAM_OFLW0: bit 4
Platform			
DMA	Peripheral RAM	SEC/DED	MEMU_PERIPH_RAM_OFLW0: bit 5
Flash			
Flash	Flash	SEC/DED	MEMU_FLASH_OFLW0: bit 0

[1] Cache tag errors can be single-bit or double-bit. Cache data array errors are reported as single-bit errors only, regardless of the number of errors reported, since the logic does not exist in the cache to distinguish them.

7.12.2.2 Uncorrectable error in flash memory can cause uncorrectable error in SRAM

Be aware of the following scenario where an uncorrectable error in flash memory can cause an uncorrectable error in SRAM to be reported. Each error triggers a separate FCCU response.

1. An uncorrectable error in flash memory occurs, and C55FMC_MCR[EER] sets to 1.
2. MEMU_FLASH_UNCERR_STS[VLD] sets to 1, and
   MEMU_ERR_FLAG[F_UCE] sets to 1.
   MEMU_FLASH_UNCERR_ADDR[ERR_ADD] records the address on which the error was detected.
3. The FCCU responds to a fault on NCF[22].

When the core register field E2ECTL0[DECCEN] is 1, these events follow:

1. The detected error in flash memory results in an exception/machine check.
2. MEMU_SYS_RAM_UNCERR_STS[VLD] sets to 1, and
   MEMU_ERR_FLAG[SR_UCE] sets to 1.
   MEMU_SYS_RAM_UNCERR_ADDR[ERR_ADD] records the address on which the error was detected.
3. The FCCU responds to a fault on NCF[16].

7.12.2.3 FlexCAN RAM error address must be shifted left twice

When a correctable or uncorrectable error occurs in a FlexCAN module's peripheral RAM, the error address captured in the MEMU_PERIPH_RAM_CERR_ADDRn or MEMU_PERIPH_RAM_UNCERR_ADDRn register is incorrect. Shift the captured address left twice to determine the actual address.

7.12.3 Fault Collection and Control Unit (FCCU) configuration

7.12.3.1 Fault inputs

Table 7-33 shows the source of the fault signals and the type of fault input to which these signals are connected at the FCCU..

All faults are marked as non-critical faults (NCF) input at the FCCU. FCCU can set reaction of each faults as critical or non-critical faults reaction. Unmasked faults with the time-out disabled are treated like critical faults by the FCCU. Table 7-32 explains the conventions used in the "Set/Clear injection" column of Table 7-33.

Table 7-32. Set/Clear injection conventions

Convention	Explanation
—	Set/clear is not supported by writing to FCCU NCF Fake Register (FCCU_NCFF)
Clear	Can be cleared only, not set, by writing to FCCU NCF Fake Register (FCCU_NCFF)
X	Can be set and cleared by writing to FCCU NCF Fake Register (FCCU_NCFF)
X (by ADC itself)	Can be set/cleared by writing to STCR2[SERR] of the applicable ADC instead of writing to FCCU NCF Fake Register (FCCU_NCFF)
X (by PMC itself)	Can be set/cleared by writing to the applicable bit of PMC_FIR instead of writing to FCCU NCF Fake Register (FCCU_NCFF)

Table 7-33. FCCU Non-Critical Faults Mapping

Non-Critical Fault	Source	Signal Description	Set/Clear injection[1]	Fault enabled	Time-out enabled
NCF[0]	Temp Sens 0/1	Temperature out of range 0/1	X (by PMC itself)	X	X
NCF[1]	Reserved		—	—	—
NCF[2]	PMC	LVD Triggered	X (by PMC itself)	X	X
NCF[3]	PMC	HVD Triggered	X (by PMC itself)	X	X
NCF[4]	PMC	Self Test Event Fault from PMC	X (by PMC itself)	X	X
NCF[5]	DCF/SSCM	Parity configuration error during reset while loading the Safe DCF Clients (DCF client with SPRD ADDR enabled, Triple Voting enabled)	—[2]	X	X
NCF[6]	SSCM / flash memory	Configuration error during reset while loading initial device configuration. ORed with flash memory reset error.	—	X	X
NCF[7]	STCU2	Critical fault indication from STCU in case LBIST or MBIST control signals go to wrong condition during user application	X	X	X
NCF[8]	STCU2	Critical fault indication from STCU	X	X	X
NCF[9]	STCU2	Non-critical Fault indication from STCU	X	X	X
NCF[10]	JTAGC_NPC_MON	Indication that Debug Modules have become active during functional mode	X	X	X
NCF[11]	RCCU_0a	Core redundancy mismatch: interface (other than D-MEM or DMA) out of lockstep	X	X	X

Table continues on the next page...

Table 7-33. FCCU Non-Critical Faults Mapping (continued)

Non-Critical Fault	Source	Signal Description	Set/Clear injection[1]	Fault enabled	Time-out enabled
NCF[12]	RCCU_0b	Core redundancy mismatch: D-MEM array interface out of lockstep [3]	X	X	X
NCF[13]	RCCU_1	Core redundancy mismatch: DMA array interface out of lockstep[3]	X	X	X
NCF[14][4]	SWT_0	Software watchdog timer reset request	—	X	X
NCF[15]	MEMU	System RAMs correctable ECC error	—	X	X
NCF[16]	MEMU	System RAMs uncorrectable ECC error	—	X	X
NCF[17]	MEMU	System RAMs error overflow (ORing of all overflows)	—	X	X
NCF[18]	MEMU	Peripheral RAMs correctable ECC error	—	X	X
NCF[19]	MEMU	Peripheral RAMs uncorrectable ECC error	—	X	X
NCF[20]	MEMU	Peripheral RAMs error overflow (ORing of all overflows)	—	X	X
NCF[21]	MEMU	Flash correctable ECC error	—	X	X
NCF[22]	MEMU	Flash uncorrectable ECC error	—	X	X
NCF[23]	MEMU	Flash error overflow (ORing of all overflows)	—	X	X
NCF[24]	PLL_0	Loss of lock	—	X	X
NCF[25]	PLL_1	Loss of lock	—	X	X
NCF[26][5]	CMU_0	XOSC vs. IRCOSC clock frequency out of range	—	X	X
NCF[27]	CMU_0	Motor clock frequency out of range	—	X	X
NCF[28]	CMU_1	Core frequency out of range	—	X	X
NCF[29]	CMU_2	PBRIDGE frequency out of range	—	X	X
NCF[30]	CMU_3	ADC clock frequency out of range	—	X	X
NCF[31]	CMU_4	SENT frequency out of range	—	X	X
NCF[32]	SIUL2	Error input pin of this device. Because of the lack of a dedicated pin for this purpose, SIUL2 External Interrupt_0 can be used to indicate if any other MCU is going into Error state. Note: If interrupt is configured as the reaction of this NCF then the interrupt controller may observe two interrupts one due to FCCU and another due to the SIUL IRQ[0]. So, appropriate configuration must be done.	—	X	X
NCF[33]	PFLASH and embedded flash memory	Address Encode Error ORed with voltage and current error of flash memory array.	Clear	X	X

Table continues on the next page...

Table 7-33. FCCU Non-Critical Faults Mapping (continued)

Non-Critical Fault	Source	Signal Description	Set/Clear injection[1]	Fault enabled	Time-out enabled
NCF[34]	PFLASH	Error in the ECC correction logic through an EDC	X	X	X
NCF[35]	PFLASH	Alarm indicating the flash memory controller detected an error in the address ECC manipulation logic through an EDC	Clear	X	X
NCF[36]	PFLASH	Alarm indicating the flash memory controller detected a transaction monitor mismatch when compared to the flash safety feedback outputs	Clear	X	X
NCF[37]	PFLASH	Alarm indicating the flash memory controller detected a transaction monitor mismatch in the pseudo-replicated calibration evaluation hardware	Clear	X	X
NCF[38]	XBAR	XBAR transaction monitor mismatch	Clear	—	X
NCF[39]	PRAMC	System RAM controller alarm	Clear	—	X
NCF[40]	TCU DFT0	Unexpected Test Mode Activation	—	—	X
NCF[41]	TCU DFT1	Unexpected Test Mode Activation	—	—	X
NCF[42]	TCU DFT2	Unexpected Test Mode Activation	—	—	X
NCF[43]	TCU DFT3	Unexpected Test Mode Activation	—	—	X
NCF[44]	Core	Safety Core Exception indication - Machine check condition	X	—	X
NCF[45]	Lockstep	Indication of disablement of Checker Core and DMA as well as RCCUs[6]	X	—	X
NCF[46]	MC_RGM	Safe mode request	—	—	X
NCF[47]	ADC_0_CF	Internal self test (critical fault)	X (by ADC itself)	—	X
NCF[48]	ADC_1_CF	Internal self test (critical fault)	X (by ADC itself)	—	X
NCF[49]	ADC_2_CF	Internal self test (critical fault)	X (by ADC itself)	—	X
NCF[50]	ADC_3_CF	Internal self test (critical fault)	X (by ADC itself)	—	X
NCF[51]	ADC_0_NCF	Internal self test (non-critical fault)	X (by ADC itself)	X	X
NCF[52]	ADC_1_NCF	Internal self test (non-critical fault)	X (by ADC itself)	X	X
NCF[53]	ADC_2_NCF	Internal self test (non-critical fault)	X (by ADC itself)	X	X
NCF[54]	ADC_3_NCF	Internal self test (non-critical fault)	X (by ADC itself)	X	X
NCF[55]	Reserved	Reserved	—	—	—
NCF[56]	INTC_MON_0	INTC Monitor Timeout	X	X	X
NCF[57]	INTC_MON_1	INTC Monitor Timeout	X	X	X

Table continues on the next page...

Table 7-33. FCCU Non-Critical Faults Mapping (continued)

Non-Critical Fault	Source	Signal Description	Set/Clear injection[1]	Fault enabled	Time-out enabled
NCF[58]	INTC_MON_2	INTC Monitor Timeout	X	X	X
NCF[59]	SIPI/DMA/Ethernet concentrator	SIPI_DMA_Ethernet concentrator transaction monitor mismatch	Clear	—	X
NCF[60]	MBIST: D-cache	Multi-bit failure indication from D-cache and D-cache tag memory cuts when MBIST is run	X	—	X
NCF[61]	MBIST: I-cache	Multi-bit failure indication from I-cache and I-cache tag memory cuts when MBIST is run	X	—	X
NCF[62]	MBIST: D-MEM	Multi-bit failure indication from Multiple D-MEM memory cuts when MBIST is run	X	—	X
NCF[63]	MBIST: SRAM	Multi-bit failure indication from Multiple system RAM memory cuts when MBIST is run	X	—	X
NCF[64]	MBIST: peripherals	Multi-bit failure indication from Multiple CAN, FlexRay, Ethernet, and DMA memory cuts when MBIST is run	X	—	X
NCF[65]	I-cache	The Control signals, which the Memory-controller sends to Memory for any operation, are latched and sent back from the Memory to Memory-Controller. The Memory-Controller compares these received control signals with the transmitted control signals. If these mismatch then the feedback alarm is generated.	Clear	—	X
NCF[66]	D-cache	The Control signals, which the Memory-controller sends to Memory for any operation, are latched and sent back from the Memory to Memory-Controller. The Memory-Controller compares these received control signals with the transmitted control signals. If these mismatch then the feedback alarm is generated.	Clear	—	X
NCF[67]	DTCM	The Control signals, which the Memory-controller sends to Memory for any operation, are latched and sent back from the Memory to Memory-Controller. The Memory-Controller compares these received control signals with the transmitted control signals. If these mismatch then the feedback alarm is generated.	Clear	—	X
NCF[68]	DMA	The Control signals, which the Memory-controller sends to Memory for any operation, are latched and sent back from the Memory to Memory-Controller. The Memory-	Clear	—	X

Table continues on the next page...

Table 7-33. FCCU Non-Critical Faults Mapping (continued)

Non-Critical Fault	Source	Signal Description	Set/Clear injection[1]	Fault enabled	Time-out enabled
		Controller compares these received control signals with the transmitted control signals. If these mismatch then the feedback alarm is generated.			
NCF[69]	RCCU_2	Redundancy mismatch: DSMC D-MEM out of lockstep [3]	X	X	X
NCF[70]	Flash memory	Flash memory low power entry error: failure to enter Stop mode upon Stop mode entry request	X	—	X
NCF[71]	SMPU	SMPU transaction monitor mismatch	X	—	X
NCF[72]	SWT_0	First timeout interrupt request from Software Watchdog of Safety Core	—	X	X
NCF[73]	PMC DCF	Digital PMC initialization error during DCF data load (status is cleared if the fault is not persistent)	—	X	X
NCF[74]	FCCU DCF	Misconfiguration of error_out pin interface of the FCCU after reset (overwriting the respective configuration bits resolves this error)	—	X	X

1. See Table 1 for an explanation of this column's contents.
2. Function not available on module
3. This fault is not generated in debug mode because the RCCUs are disabled in debug mode. See Core lockstep and RCCU disablement in debug mode for more information.
4. Always configure the FCCU reaction to NCF[14] as a short or long functional reset. Never configure it as an interrupt.
5. Always configure the FCCU reaction to NCF[26] as a short or long functional reset. Never configure it as an interrupt.
6. If RCCUs are disabled due to entry to debug mode, this fault is not triggered. If RCCU disablement occurs for another reason, this fault is triggered. See Core lockstep and RCCU disablement in debug mode for more information about RCCU disablement in debug mode.

NOTE

Functional reset is not generated by any of the fault by default (after reset).

7.12.3.2 EOUT[0:1] and FCCU_F[0:1] signals

The external signals generated by the FCCU have different names at the module level and the chip level. They are named:

- EOUT[0:1] (fcc_eout[0:1]) at the module level, such as in the dedicated FCCU chapter.
- FCCU_F[0:1] at the chip level, such as in the Signal Description chapter.

7.12.3.3 FCCU chip-specific register reset values

This table lists in alphabetical order by mnemonic the FCCU registers whose reset values are specific to this chip. (The reset values for all other FCCU registers are specified in Register descriptions.)

Register	Reset value
FCCU_CFG	Loaded from FCCU record after reset
FCCU_CFG_TO	0000_0006h
FCCU_NCF_CFG0	FFFF_FFFFh
FCCU_NCF_CFG1	FF7F_FFFFh
FCCU_NCF_CFG2	0000_07FFh
FCCU_NCF_E0	FFFF_FFFFh
FCCU_NCF_E1	0778_003Fh
FCCU_NCF_E2	0000_0720h
FCCU_NCF_TO	0000_FFFFh
FCCU_NCF_TOE0	FFFF_FFFFh
FCCU_NCF_TOE1	FF7F_FFFFh
FCCU_NCF_TOE2	0000_07FFh
FCCU_NCFS_CFG0	0000_0000h
FCCU_NCFS_CFG1	0000_0000h
FCCU_NCFS_CFG2	0000_0000h
FCCU_NCFS_CFG3	0000_0000h
FCCU_NCFS_CFG4	0000_0000h

7.12.3.4 FCCU chip-specific glitch-filter guidelines

In this chip, the FCCU glitch filter is assigned to the NCF1 channel; however, this channel is tied low instead of being connected to a pin on the boundary of the chip.

Ignore the FCCU_CTRL[FILTER_BYPASS] and FCCU_CTRL[FILTER_WIDTH] fields because they have no effect on anything.

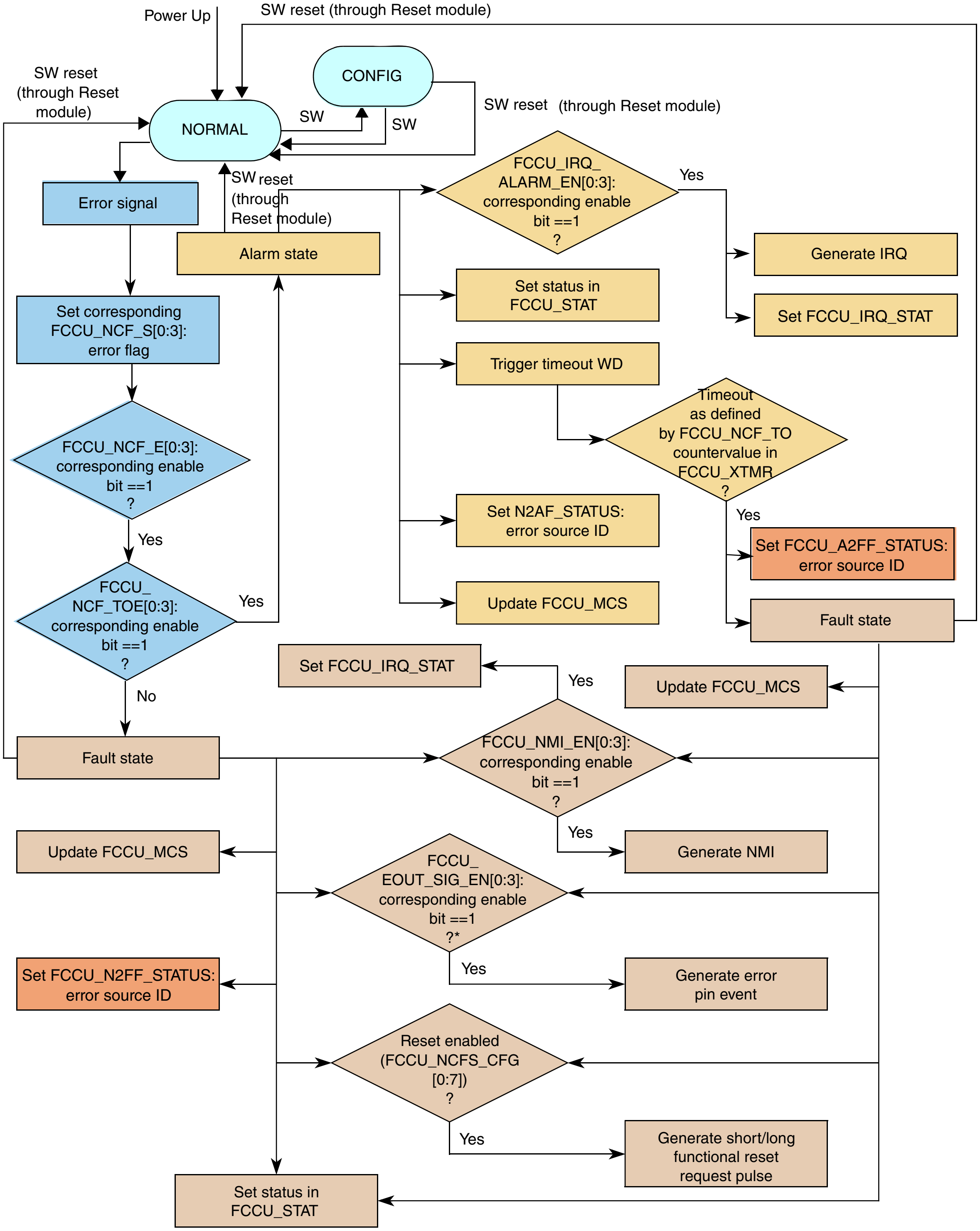
7.12.3.5 FCCU_STAT[PhysicErrorPin] reset value

The reset value of the FCCU_STAT[PhysicErrorPin] depends on the current state of the physical error pin. The value also depends on the selected protocol. See EOUT interface.

7.12.3.6 FOSU_COUNT value

The FCCU Output Supervision Unit (FOSU) has a timer whose duration is the value FOSU_COUNT. On this chip, the value is 65,535 IRCOSC clock cycles.

7.12.3.7 Error signal flow diagram



* This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

Figure 7-8. Error signal flow

7.12.4 Self-Test Control Unit (STCU2) configuration

See the Self-Test Control Unit (STCU2) details.

7.12.5 Register protection (REG_PROT) configuration

7.12.5.1 List of protected registers

An appendix of this document provides a comprehensive list of protected registers on the device.

7.13 Debug

The device provides Nexus with Reduced Port Mode (RPM) and the Aurora interface (FPM) for debug access.

The Aurora interface allows the Nexus protocol information to be transmitted serially at a high rate of speed over two Aurora lanes.

See the Debug details for more information.

Table 7-34. PRN for existing Masksets

Maskset name	Part Revision Number(PRN)
1N15P	"0x3"
1N65H	"0x1"

7.14 Wakeup Unit (WKPU) configuration

The WKPU outputs an NMI source signal that triggers an interrupt connected to the NMI input of the FCCU, which generates a core wakeup signal if the chip is in stop mode or provides an NMI in all other cases.

7.14.1 Chip-specific WKPU register reset values

The following table lists WKPU register reset values that are specific to this chip.

Table 7-35. Chip-specific WKPU register reset values

Register	Reset value	Notes
NCR	0000_0000h	NDSS0 field resets to 00b

Chapter 8
Reset

8.1 Introduction

The chapter describes the global and local reset behavior of the device.

8.1.1 Global Reset

The global reset behavior refers to device reset behavior controlled by the MC_RGM which affects several modules of the device at the same time. The global reset is linked to the reset process described in Section 8.2, Global Reset Process.

8.1.2 Local Reset

The local reset behavior refers to module reset behavior controlled by the MC_RGM or dedicated modules which affects only one module at a time. Local resets are described in the module chapters.

8.2 Global Reset Process

This section describes the global reset process. Throughout this section the term *global* is omitted.

8.2.1 Overview

The global reset process is a sequence of several reset phases. Each reset phase has a specific entry condition, a specific exit condition, and a specific device reset behavior, which is different among the reset phases. The reset phases are executed in a specific order, thus building the actual global reset process. There are several modules contributing to the global reset process. All modules of the device can be affected by the global reset process.

8.2.2 Reset Process Modules

All modules which can enter, gate, or control the reset process are listed in Table 8-1 together with their type of contribution. The types of contribution are described in Table 8-2.

Table 8-1. Reset Process Modules

Module	Contribution Type		
	event	gating	control
Section 8.2.2.1, Reset Generation Module (MC_RGM)	v	v	v
Section 8.2.2.2, Power Management Controller (PMC)	v	v	v
Section 8.2.2.3, Mode Entry Module (MC_ME)	v		
Section 8.2.2.4, System Status and Configuration Module (SSCM)		v	
Section 8.2.2.5, Self-Test Control Unit (STCU)			v
Section 8.2.2.6, Fault Collection and Control Unit (FCCU)	v	v	
Section 8.2.2.7, Embedded Flash Memory (c55fmc)	v	v	
Section 8.2.2.8, JTAG Controller (JTAGC)	v		

Table 8-2. Module Contribution Type

Contribution Type	Description
event	The module may generate events which may trigger the entry into an reset state, if the event configuration in the MC_RGM allows this.
gating	The module can prevent the exit of a reset state if it has not reached a certain internal state.
control	The module controls the sequence of the reset process.

8.2.2.1 Reset Generation Module (MC_RGM)

The MC_RGM is one of three controlling modules for the reset process. The MC_RGM can receive events from other modules and starts the reset process based on the configuration for these events. The event configuration defines the reset phase at which the reset process starts. The MC_RGM also controls the reset process sequence based on the gating information it receives from the gating modules.

8.2.2.2 Power Management Controller (PMC)

The PMC provides two types of signals for the reset process control.

As long as the supply voltages are not reliable, the PMC provides signals to put the whole device into a power on reset state.

As long as the supply voltages are reliable but above or below certain operational limits, the PMC provides signals to allow the device to enter the reset process.

8.2.2.3 Mode Entry Module (MC_ME)

The MC_ME can generate events to enter the reset process. These events can be triggered only by application software.

8.2.2.4 System Status and Configuration Module (SSCM)

The SSCM is started automatically during a specific phase of the reset process. The SSCM fetches DCF records from the flash memory and distributes the read values via an internal DCF bus to the related modules. As long as the SSCM is still fetching DCF records, this specific reset phase cannot be exited.

8.2.2.5 Self-Test Control Unit (STCU)

The STCU controls the execution of the startup and shutdown self-test. Depending on its configuration, the STCU is started when PHASE3[DEST] is exited or when software triggers the self-test. The SCTU generates a functional event when the self-testing has been completed.

8.2.2.6 Fault Collection and Control Unit (FCCU)

The FCCU generates reset events and gates the reset process while it is configuring itself based on the DCF record content.

The FCCU's FOSU also generates a reset event when a fault occurs but the FCCU fails to react to that fault.

8.2.2.7 Embedded Flash Memory (c55fmc)

Failure of flash memory initialization, which generates a reset event and gates the reset process, usually results from misconfigured data for self-test or trimming.

8.2.2.8 JTAG Controller (JTAGC)

JTAGC TAP controller generates this event to allow the external debugger to reset the SoC via a JTAG command (such as JTAG EXTTEST) without using the external RESET_B pin.

8.2.3 Reset Process Sequence

This section specifies the reset process sequence. An overview of the reset process sequence appears in the following figure. When the reset process has reached the IDLE state, the device is operational.

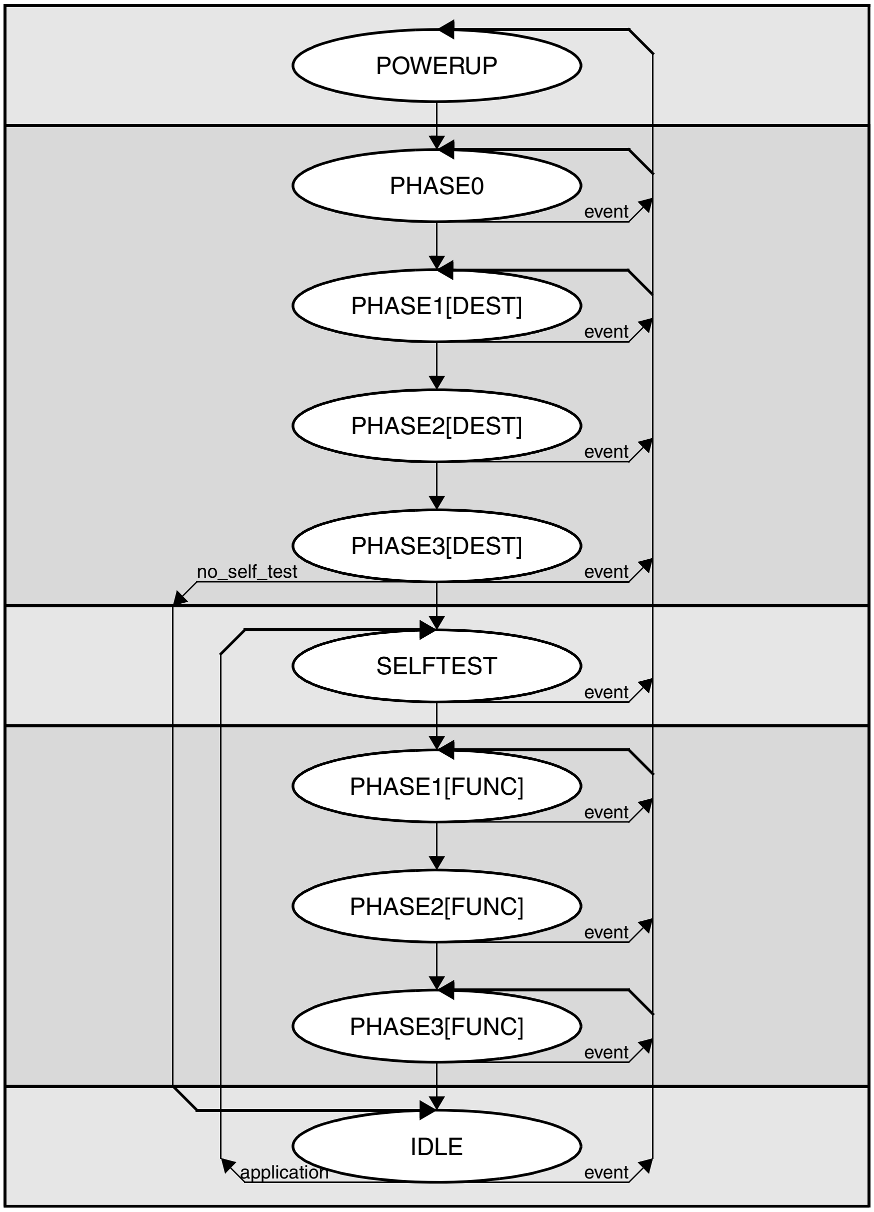


Figure 8-1. Reset Process Sequences

8.2.4 Reset Process State Transitions

This section specifies the transitions between the reset states in the reset process. For each state, the state entry, the state execution, and the state exit are described.

8.2.4.1 POWERUP

8.2.4.1.1 POWERUP State Entry

This state is entered from any state if at least one of the following events occurs:

- Power is applied to the unpowered device.
- PMC has detected a low power condition that is lower than the minimum operating voltage
- EXT_POR_B pin is driven low

8.2.4.1.2 POWERUP State Execution

During this state the PMC is waiting for stable power supply.

All output PADs are driven to known values, and all bidirectional PADs are configured as inputs.

8.2.4.1.3 POWERUP State Duration

There are no bounds on the duration of this state.

8.2.4.1.4 POWERUP State Exit

Table 8-3. POWERUP Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	(1) The EXT_POR_B pin is high and (2) the PMC module indicates all supplies are in the operational range	IRCOSC MC_RGM

8.2.4.2 PHASE0

8.2.4.2.1 PHASE0 State Entry

This state can be entered from

- all other reset states when the state exit condition is present.

8.2.4.2.2 PHASE0 State Execution

All trimming bits are reset to their default values.

All output PADs are driven to known values, and all bidirectional PADs are configured as inputs.

8.2.4.2.3 PHASE0 State Exit

Table 8-4. PHASE0 Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE1[DEST]	(1) The MC_RGM indicates that all sources of enabled destructive reset events are deasserted, and (2) the IRCOSC module indicates the generation of a stable clock, and (3) the PMC module has ***deasserted dpmc_rgm_phase0_lvd_b***.	

8.2.4.3 PHASE1[DEST]

8.2.4.3.1 PHASE1[DEST] State Entry

This state can be entered from state

- PHASE0,
- PHASE1[FUNC],
- PHASE2[FUNC],
- PHASE3[FUNC],
- IDLE.

8.2.4.3.2 PHASE1[DEST] State Execution

There are no state changes in any module; only a counter in the MC_RGM is running.

8.2.4.3.3 PHASE1[DEST] State Exit

Table 8-5. PHASE1[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	

Table continues on the next page...

Table 8-5. PHASE1[DEST] Next State Table (continued)

Next State	Condition	Process/Module Started at Exit
PHASE2[DEST]	8 IRCOSC clocks passed by after state entry.	FLASH - starts initialization sequence FCCU - enters configuration state in preparation for receiving DCF record content STCU - waits for configuration via DCF records

8.2.4.4 PHASE2[DEST]

8.2.4.4.1 PHASE2[DEST] State Entry

This state can be entered from state

- PHASE1[DEST].

8.2.4.4.2 PHASE2[DEST] State Execution

FLASH executes an initialization sequence for slow array access.

8.2.4.4.3 PHASE2[DEST] State Exit

Table 8-6. PHASE2[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE3[DEST]	The FLASH module indicates the availability of slow array access.	SSCM - starts device configuration and boot location search

8.2.4.5 PHASE3[DEST]

8.2.4.5.1 PHASE3[DEST] State Entry

This state can be entered only from state

- PHASE2[DEST].

8.2.4.5.2 PHASE3[DEST] State Execution

SSCM runs device configuration and boot location search.

The FCCU executes its configuration sequence.

The PMC adjusts its LVD and HVD levels according to the DCF record content, adjusts its output to the full voltage level, and executes its self-test.

The FLASH transitions from slow to full speed array access when full voltage has been achieved by the PMC.

8.2.4.5.3 PHASE3[DEST] State Exit

Table 8-7. PHASE3[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
SELFTEST	(1) The FLASH module indicates availability of full speed array access, and (2) the SSCM module indicates device configuration is done, and (3) the FCCU module indicates it has completed its configuration sequence, and (4) the PMC module indicates it has finished its internal self-test, and (5) the STCU module is configured to execute a startup self-test.	BIST on - FCCU - CORE - PERIPHERALS
IDLE	(1) the FLASH module indicates availability full speed array access, and (2) the SSCM module indicates device configuration is done, and (3) the FCCU module has completed its configuration sequence, and (4) the PMC module indicates it has finished its internal self-test.	CORE ALL OTHER PERIPHERALS

8.2.4.6 SELFTEST

8.2.4.6.1 SELFTEST State Entry

This state can be entered only from state

- PHASE3[DEST],
- IDLE.

8.2.4.6.2 SELFTEST State Execution

SELFTEST is running on FCCU, CORE, PERIPHERALS.

8.2.4.6.3 SELFTEST State Exit

Table 8-8. SELFTEST Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[FUNC]	STCU module indicates end of self test.	

8.2.4.7 PHASE1[FUNC]

8.2.4.7.1 PHASE1[FUNC] State Entry

This state can be entered from states

- SELFTEST
- PHASE3[FUNC]
- IDLE.

8.2.4.7.2 PHASE1[FUNC] State Execution

There are no state changes in any module; only a counter in the MC_RGM is running.

8.2.4.7.3 PHASE1[FUNC] State Duration

The duration of this state is 8 IRCOSC clock cycles.

8.2.4.7.4 PHASE1[FUNC] State Exit

Table 8-9. PHASE1[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	When self test is enabled: The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is an external reset.	
PHASE1[FUNC]	The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is not an external reset. or When self test is disabled: The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is an external reset.	
PHASE2[FUNC]	(1) The MC_RGM indicates that all sources of enabled long functional reset events are deasserted, and (2) 8 IRCOSC clocks passed by after state entry.	FLASH - starts initialization sequence - enters configuration state in preparation for receiving DCF record content if PHASE1[FUNC] was entered on the exit of the SELFTEST state

8.2.4.8 PHASE2[FUNC]

8.2.4.8.1 PHASE2[FUNC] State Entry

This state can be entered from state

- PHASE1[FUNC].

8.2.4.8.2 PHASE2[FUNC] State Execution

FLASH executes an initialization sequence for slow array access.

8.2.4.8.3 PHASE2[FUNC] State Exit

Table 8-10. PHASE2[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	When self test is enabled: The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is an external reset.	
PHASE1[FUNC]	The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is not an external reset. or When self test is disabled: The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is an external reset.	
PHASE3[FUNC]	The FLASH module indicates the availability of slow array access.	SSCM - starts device configuration

8.2.4.9 PHASE3[FUNC]

8.2.4.9.1 PHASE3[FUNC] State Entry

This state can be entered from states

- PHASE2[FUNC]
- IDLE.

8.2.4.9.2 PHASE3[FUNC] State Execution

SSCM runs device configuration if the state was entered from PHASE2[FUNC].

The FCCU executes its configuration sequence if PHASE3[FUNC] was entered via the SELFTEST state.

The FLASH makes the transition from slow to full speed array access.

8.2.4.9.3 PHASE3[FUNC] State Exit

Table 8-11. PHASE3[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	When self test is enabled: The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is an external reset.	
PHASE1[FUNC]	The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is not an external reset. or When self test is disabled: The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is an external reset.	
IDLE	(1) The FLASH module indicates availability of full speed array access, and (2) the SSCM module indicates device configuration is done, and (3) the FCCU module has completed its configuration sequence, and (4) the PMC module indicates it has finished its internal self-test.	CORE ALL OTHER PERIPHERALS

8.2.4.10 IDLE

8.2.4.10.1 IDLE State Entry

The IDLE state can be entered only from states

- PHASE3[DEST]
- PHASE3[FUNC].

8.2.4.10.2 IDLE State Execution

During the IDLE state, the device is fully operational.

8.2.4.10.3 IDLE State Duration

There are no bounds on the duration of this state.

8.2.4.10.4 IDLE State Exit

The IDLE state next state table is given in the following table.

Table 8-12. IDLE Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	When self test is enabled: The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is an external reset.	
PHASE1[FUNC]	The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is not an external reset. or When self test is disabled: The MC_RGM has detected that the source of the enabled long functional reset event is asserted, and the source is an external reset.	
PHASE3[FUNC]	The MC_RGM has detected that the source of an enabled short functional reset event is asserted.	
SELFTEST	The application has triggered the start of the shutdown self-test.	

8.2.5 Module Status During Reset Process

The following table specifies the status of the device modules during the reset process. The status abbreviations are explained in Table 8-14.

Table 8-13. Module status during reset states

Module	POWERUP	Global Reset State								
		PHASE0	PHASE1[DEST]	PHASE2[DEST]	PHASE3[DEST]	SELFTEST	PHASE1[FUNC]	PHASE2[FUNC]	PHASE3[FUNC]	IDLE
PMC	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON
IRCOSC[1]	RST	ON	ON	ON	ON	ON	ON	ON	ON	ON
MC_RGM	RST	ON	ON	ON	ON	ON	ON	ON	ON	ON
XOSC[2]	RST	RST	RST	RST	ON	ON	ON	ON	ON	ON
FLASH	RST	RST	RST	ON	ON	ON	RST	ON	ON	ON
SSCM	RST	RST	RST	RST	ON	ON	RST	RST	ON	ON
FCCU	RST	RST	RST	ON	ON	BIST	ON	ON	ON	ON
STCU	RST	RST	RST	ON	ON	ON	ON	ON	ON	ON
CORE	RST	RST	RST	RST	RST	BIST	RST	RST	RST	ON
OTHERS	RST	RST	RST	RST	RST	BIST	RST	RST	RST	ON

1. IRCOSC registers are reset with the PHASE3 reset, but the analog portion of the IRCOSC is reset only with the POWERUP. Therefore, as the table shows, the IRCOSC is running during all phases except POWERUP.
2. XOSC registers are reset with the PHASE3 reset, but the analog portion of the XOSC is reset during the PHASE3 reset sequence (pre-self-test) only if configured in the UTEST portion of the flash memory.

Table 8-14. Module Status Glossary

Module Status	Status Description
RST	Module is held in reset by the global reset process
BIST	Module is being tested or held in a non-functional state during self-test execution as controlled by the STCU
ON	Module is functional

Chapter 9
Device Configuration Format (DCF) Records

9.1 Introduction

The system reset sequence is a complex process implementing a considerable amount of initialization before releasing reset.

Before the device can be used by an application, the application code, application specific reset values, and Device Configuration Format (DCF) records must be properly programmed into their respective flash memories. DCF records are used to configure certain registers – the DCF clients – in the device during the reset sequence. For this purpose, the System Status and Control Module (SSCM) reads the DCF records from flash and writes the configuration information to the specified DCF clients. Thus, many operational aspects of the device are properly configured before the reset signal is released and normal device operation begins.

9.2 DCF clients

DCF clients are simply designated registers in the device which receive their value from DCF records. (Some DCF clients may additionally be writable directly from the CPU, once the device has exited the reset sequence. In this case the DCF records placed in UTEST will initialize the client, i.e. defining it's reset value.)

Depending on their usage DCF clients may be equiped with a variety of safety features and modification rules.

9.2.1 Safety features of DCF clients

Depending on its role in the device, a client may be equipped with a single safety feature or a combination of safety features.

9.2.1.1 Parity

If a DCF client implements parity checking, then it will receive a parity bit additionally to it's data in the DCF record. During device operation it will continuously monitor whether the data it's stores matches the parity and report errors to the SSCM.

9.2.1.2 Triple voting

DCF clients that use the triple voting have three copies of the register. The SSCM will write to all three registers in a single write cycle. During device operation the DCF client continuously monitors whether all three copies match and reports errors to the SSCM. The device will use the majority result, so single errors will not affect device operation.

9.2.1.3 Spread triple voting

DCF clients that use spread triple voting have three copies of the register which are individually addressed and where the data is stored in three different formats. The SSCM will fetch these records independently. During device operation the DCF client continuously monitors whether all three copies can be resolved from the stored format to the the value and reports errors to the SSCM. The device will use the majority result, so single errors will not affect device operation.

9.2.1.4 Second write check

This feature is used by DCF clients which receive their records twice - e.g. once before and once after BIST for clients which retain their value during self test. They will monitor whether the second write matches the first and report deviations to the SSCM.

9.2.2 DCF client modification rules

Depending on its role in the device a client may implement one or a combination of modification rules. If a modification rule is in effect, the order in which DCF records are placed in the record list may be important.

9.2.2.1 Write Once

A DCF client using the Write Once rule can only be written with a single DCF record. Records appended later in the list will be ignored and not change the value of the client.

9.2.2.2 Write 0 only

A bit in a DCF client can only be written from a 1 to a 0. So, if an earlier DCF record has set a bit to 0, then an attempt by a later record to set this bit to a 1 will be ignored.

9.2.2.3 Write 1 only

A bit in a DCF client can only be written from a 0 to a 1. So, if an earlier DCF record has set a bit to 1, then an attempt by a later record to set this bit to a 0 will be ignored.

9.3 DCF records

An individual DCF record consists of information to locate the corresponding DCF client internal to the device (control word) and the data to be written to that client (data word).

DCF records appear as contiguous series of entries programmed at a specific address within flash memory. The following structure must be present:

- The first DCF record must be a start record. This record must be placed at the beginning of a DCF area in flash memory to indicate to the device that the following records must be processed.
- DCF records containing configuration data must immediately follow the start record with no blank records between-an unprogrammed record is interpreted as a stop record and no DCF records following that record are processed. This allows one to program the records in several sessions, each time appending new records at the end of the list.
- The end of the configuration records are indicated by the presence of a stop record; indicated by the stop bit being set. Normally, the stop bit is never set in a valid UTEST DCF record programmed during production, since this would prevent appending additional UTEST records. The flash memory location following the last UTEST DCF record programmed at the factory is an unprogrammed location. Therefore, that location's contents will be 0xFFFF_FFFF. Thus, the stop bit location in this unprogrammed flash memory location will be a logic 1, signifying that this is the last DCF record and it is not to be acted upon.

For a detailed discussion of the basic DCF record format, its fields, and specific records (start, stop) refer to the corresponding chapter in the documentation of the System Status and Control Module (SSCM).

9.4 DCF clients available in the SOC

This section describes user-visible DCF clients in the device. These clients can be controlled by programming DCF records in UTEST. The default values given are those which are active if there are no records for these clients in UTEST.

Table 9-1. Clients which can be set by the user.

<table>
<tr><th>DCF_Client</th><th>Control Word</th><th>Default Value</th><th>Modification Rule</th><th>Safety</th><th>Description</th></tr>
<tr><td>pwd_low</td><td>0x0100_0004</td><td>0xFFFF_FFFF</td><td>standard</td><td>-</td><td>Flash Backdoor
Password Low</td></tr>
<tr><td>pwd_hi</td><td>0x0100_0008</td><td>0xFFFF_FFFF</td><td>standard</td><td>-</td><td>Flash Backdoor
Password High</td></tr>
<tr><td>serial_and_sec</td><td>0x0100_000C</td><td>0x55AA_55AA</td><td>standard</td><td>-</td><td>Serial Boot Mode and Censorship</td></tr>
<tr><td>otp_mid_low</td><td>0x0100_0010</td><td>0x8000_0000</td><td>standard</td><td>-</td><td>OTP - Mid and Low Blocks</td></tr>
<tr><td>otp_high</td><td>0x0100_0014</td><td>0x0000_0000</td><td>standard</td><td>-</td><td>OTP - High Blocks</td></tr>
<tr><td>otp_256_l</td><td>0x0100_0018</td><td>0x0000_0000</td><td>standard</td><td>-</td><td>OTP - Lower 256k Blocks</td></tr>
<tr><td>soc_conf</td><td>0x0100_0024</td><td>0x0000_0001</td><td>standard</td><td>-</td><td>SOC configuration</td></tr>
<tr><td>fccu</td><td>0x0100_0104 /
0x0100_0108 /
0x0100_0110</td><td>0x0000_003F</td><td>standard</td><td>spread triple voting</td><td>FCCU Settings</td></tr>
<tr><td>STCU clients</td><td colspan="5"><i>See STCU chapter</i></td></tr>
<tr><td>PMC_REE</td><td>0x00400224/
0x00400228/
0x00400230</td><td>0x000000FF/
0xFFFFFF00/
0x8000007F</td><td>standard</td><td>spread triple voting</td><td>The 8 bits used in DCF for controlling the resets is mapped to PMC_REE_0 IPS register in PMC module as below
<table>
<tr><td>Bit number</td><td>Bit name</td></tr>
<tr><td>0</td><td>VD3RES_C</td></tr>
<tr><td>1</td><td>VD3RES_H</td></tr>
<tr><td>2</td><td>VD4RES_C</td></tr>
<tr><td>3</td><td>VD6RES_C</td></tr>
<tr><td>4</td><td>VD6RES_F</td></tr>
<tr><td>5</td><td>VD6RES_IM</td></tr>
<tr><td>6</td><td>VD6RES_ADC</td></tr>
<tr><td>7</td><td>VD6RES_O</td></tr>
</table></td></tr>
</table>

Table continues on the next page...

Table 9-1. Clients which can be set by the user. (continued)

<table>
<tr><th>DCF_Client</th><th>Control Word</th><th>Default Value</th><th>Modification Rule</th><th>Safety</th><th>Description</th></tr>
<tr><td>PMC_LVD_MISC</td><td>0x00400144/
0x00400148/
0x00400150</td><td>0xE3xxxxxx/
0x1Cxxxxxx/
0xF1xxxxxx</td><td>standard</td><td>spread triple voting</td><td>The bits used in DCF for controlling the resets is mapped to PMC_REE_TD IPS register in PMC module as below
<table>
<tr><td>Bit number</td><td>Bit name</td></tr>
<tr><td>24</td><td>Reserved</td></tr>
<tr><td>25</td><td>Reserved</td></tr>
<tr><td>26</td><td>TEMP0_0/
TEMP1_0</td></tr>
<tr><td>27</td><td>TEMP0_2/
TEMP1_2</td></tr>
<tr><td>28</td><td>TEMP0_3/
TEMP1_3</td></tr>
<tr><td>29</td><td>Reserved</td></tr>
<tr><td>30</td><td>Reserved</td></tr>
<tr><td>31</td><td>Reserved</td></tr>
</table></td></tr>
</table>

9.4.1 Security and serial boot mode records

For a description for the flash memory backdoor password, serial boot mode, and security records, see the SSCM details in Security and Serial access security.

9.4.2 OTP records

The description of the OTP records is in the Flash OTP Control details.

9.4.3 FCCU record

The FCCU record allows you to define the initial value for the FCCU_CFG register. The register holds this initial value after the device exits the reset sequence. See Figure 9-1. For a description of the register, see the dedicated FCCU chapter.

As a client with high importance for the safety concept, the FCCU DCF client uses the spread triple voting feature. This feature means that three records must be written to set the values for the client.

The meaning of the data bits is not the same for all records. The first record uses the standard format, the second inverts the values of the first (see Figure 9-2), and the third shift-rotates the values to the left (see Figure 9-3).

0:19	20	21	22	23:25	26:31
rsrvd	CM	SM	PS	FOM	FOP

0:31
0100_0104h

Figure 9-1. FCCU Record (standard)

0:19	20	21	22	23:25	26:31
rsrvd	~CM	~SM	~PS	~FOM	~FOP

0:31
0100_0108h

Figure 9-2. FCCU Record (inverted)

0:19	20	21	22	23	24:26	27:31
rsrvd	FOP(0)	CM	SM	PS	FOM	FOP(5:1)

0:31
0100_0110h

Figure 9-3. FCCU Record (rotated right)

9.4.3.1 Field descriptions

- The CM field is always 0b.
- For descriptions of the other fields, see Configuration (FCCU_CFG).

9.4.4 SOC configuration record

The SOC configuration record configures central functionality of the chip.

0:29	30	31
rsrvd	WDE	CCE

0:31
0100_0024h

Figure 9-4. SOC Configuration Record

For descriptions of the WDE and CCE fields, see SSCM_UOPS register and soc_conf DCF client.

9.4.5 STCU configuration records

The STCU provides a large number of configuration records. The STCU chapter provides details.

Chapter 10
Device Security

10.1 Device Security

Device security, which is implemented by the SSCM, focuses on preventing unauthorized access to the flash memory contents.

The following features are available:

- 64-bit password protection
- Challenge timeout to protect against brute force attacks
- Swallowed key feature
- Voltage-attack protection
- Public serial boot mode to allow limited device diagnostics without giving access to the Flash
- Private serial boot mode for application code update
- JTAG unsecure feature to allow debugging of secured devices for authorized developers

Chapter 11
Debug

11.1 Introduction

The chapter discusses the device specific information for the debug modules. This device implements both, a Nexus trace auxiliary port with MDO/MSEO pins and a Nexus Aurora interface. The Aurora interface allows the Nexus protocol information to be transmitted serially at a high rate of speed over two Aurora lanes. The Aurora protocol handles the encoding of the data and striping the information across the Aurora lanes available on the device. The device starts up in JTAG mode.

The Event Out ($\overline{\text{EVTO}}$) signal is connected to the external PAD[90] device pin and the Event In (EVTI_IN) signal is connected to the external PAD[91] device pin.

NOTE

It should be expected that during a destructive reset event, the entire device is reset including debug modules and the debugger loses connectivity with the chip. Whenever the device is under debug and gets a long functional reset, it will lose connection with the debugger. During software development if a reset cycle is necessary without disconnecting the debugger, a short functional reset can be used.

11.2 e200z4 core debug support

Internal debug support in the e200z4 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes.

For details, see e200z4 Core Debug Support.

11.3 Core lockstep and RCCU disablement in debug mode

In debug mode, the RCCUs are disabled, and the cores do not operate in Lockstep mode.

The actual enabled/disabled state of the RCCUs (including any automatic disablement by debug) can be read by software in the RCCU Status (RCCU_STAT) register.

Table 11-1. RCCU Status register details

Address	Register name	Width (in bits)	Access	Reset value
FFC0_C004h	RCCU_STAT	32	R	0000_0000h

The read-only RCCU Status register holds the RCCU Status Flag (RSF) that indicates RCCU enablement/disablement.

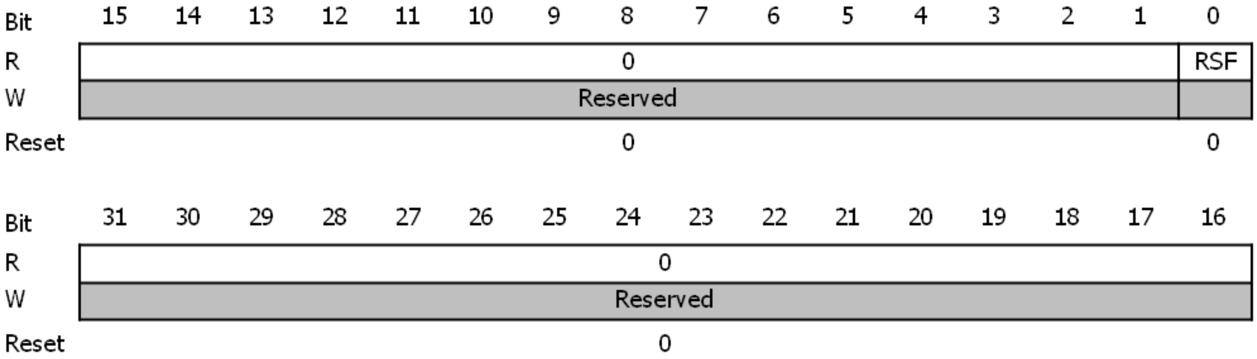


Table 11-2. RCCU_STAT field descriptions

Field	Description
0 RSF	RCCU active Status Flag 0 RCCUs are disabled. Cores are not in Lockstep mode. 1 RCCUs are enabled. Cores are in Lockstep mode.
1-31 Reserved	This read-only field is reserved and always has the value 0.

11.4 Run control and memory access

Run control is any operation required to control the device operation. Run control includes starting and stopping execution of code by the processor, as well as access to memory while the device is stopped to download code.

The JTAG interface supports both boundary scan and debug modes. It is based on the IEEE1149.1-2001 standard. The JTAG Controller (JTAGC) operates in the standard IEEE 1149.1 5-wire interface.

11.5 JTAG Controller (JTAGC)

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format. All Nexus debug registers are accessed and configured via the JTAG controller.

For details, see the JTAG Controller (JTAGC) chapter.

11.5.1 JTAGC ACCESS_AUX block instructions

The ACCESS_AUX instructions are described in this table.

Table 11-3. ACCESS_AUX 5-bit JTAG instructions

Instruction	Code[4:0]	Instruction Summary
ACCESS_AUX_x	10000–11110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
ACCESS_AUX_NPC	10000	Enables access to the NPC TAP controller
Reserved	10001	Reserved
Reserved	10010	Reserved
Reserved	10011	Reserved
Reserved	10100	Reserved
Reserved	10101	Reserved
Reserved	10110	Reserved
ACCESS_AUX_NXMC_0	10111	Enables access to the Nexus XBAR Multimaster Client 0 TAP controller
ACCESS_AUX_NXMC_1	11000	Enables access to the Nexus XBAR Multimaster Client 1 TAP controller
Reserved	11001	Reserved
ACCESS_AUX_LSM	11010	Enables access to multiple cores in parallel when in lock step mode (LSM)
Reserved	11011	Reserved
Reserved	11100	Reserved
Reserved	11101	Reserved
ACCESS_AUX_NAL	11110	Enables access to NAL

11.5.2 TEST_CTRL register

The following register details elaborate on the general description in TEST_CTRL Register of the dedicated JTAGC chapter.

TEST_CTRL

LSB=0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BSR_HYS	BSR_SRE[1:0]		BSR_PUS	BSR_PUE	BSR_PAD_CTRL_EN	(R)0[1]	(R)0[1]	(R)0[1]	NEX_RDY_PORT_EN	(R)1	(R)0[1]	(R)1[1]	(R)1[1]	(R)1[1]	(R)0[1]	(R)1[1]
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1

1. This bit is reserved. The defined value ensures future compatibility.

Table 11-5. TEST_CTRL register field descriptions

Field	Description
BSR_HYS	BSR Input Hysteresis Control Enables input hysteresis for the associated pin 0 Disabled 1 Enabled
BSR_SRE[1:0]	BSR Slew Rate Control Controls the slew rate of the output driver 00 Half drive strength with slew rate control 01 Full drive strength with slew rate control 10 Half drive strength without slew rate control 11 Full drive strength without slew rate control
BSR_PUS	BSR Pads Pull Select Selects pull-up or pull-down 0 pull-down selected 1 pull-up selected
BSR_PUE	BSR Pads Pull Enable Enables active pull-up/down 0 pull disabled 1 pull enabled
BSR_PAD_CTRL_EN	Boundary Scan Mode Pad Control Enable Enables the pad controls for all BSR pads when the JTAG EXTEST or CLAMP instruction is active. Control occurs via the PTI module. 0 BSR pad control disabled

Table continues on the next page...

Table 11-5. TEST_CTRL register field descriptions (continued)

Field	Description
	1 BSR pad control enabled
NEX_RDY_PORT_EN	RDY_B Enable Enables RDY_B functionality for corresponding GPIO pin. Setting this bit overrides the SIUL2 output muxing configuration for the corresponding pad. 0 Disables RDY_B functionality (output mux for the pad is controlled by SIUL2) 1 Enables RDY_B functionality (output mux for the pad is controlled by JTAGC)

11.5.3 Device identification register reset values

The reset value of the Device identification register (JTAG ID) Register depends on the chip version:

- Maskset N15P : 29B4501Dh
- Maskset 1N65H : 19B4501Dh
- Maskset 0N65H : 09B4501Dh

11.6 Nexus Port Controller (NPC)

11.6.1 Chip-specific NPC information

This section presents device-specific parameterization, customization, and feature availability information not specifically referenced in the remainder of this chapter.

11.6.1.1 Parameter values

The parameter values for this device are shown in the following table.

Table 11-6. Device parameter values

Parameter	Value
Number of MDO pins available (Reduced port mode)	4
Part identification number (PIN)	345h
Manufacturer identity code (MIC)	00Eh
Design center code (DC)	26h

11.6.1.2 DID register reset and parameter values

The reset value of the Nexus Device ID (DID) Register depends on the chip version:

- Maskset 1N65H: 19B4501Dh
- Maskset 0N65H: 09B4501Dh
- Maskset N15P: 29B4501Dh

The PRN, PIN, and DC parameters define the chip-specific values of correspondingly named fields in the DID register. The following table provides the values of those DID fields and of the register's MIC field.

Table 11-7. Values of PRN, DC, PIN, and MIC fields of DID register

DID field	Device-specific value
PRN	1h[1]
DC	26h
PIN	345h
MIC	00Eh

[1] For maskset 0N65H, PRN is 0h.

11.6.1.3 ACR field values

The LVDS_RXOPT and LVDS_TXOPT fields of the Aurora Configuration Register (ACR) function as controls for LVDS pads. For these fields, the following table identifies the bits actually used on the device and the meaning of their settings.

NOTE

> Bits that are not listed are reserved.

Table 11-8. Settings for LVDS_RXOPT and LVDS_TXOPT fields of ACR

Field bit	Device-specific setting descriptions
LVDS_RXOPT1	Default value is 0. When set to 1, increases RX current by 20%.
LVDS_RXOPT0	Default value is 0. When set to 1, decreases RX current by 20%.
LVDS_TXOPT7	Default value is 0. When set to 1, decreases internal bias current by 50%.
LVDS_TXOPT6	Default value is 0. When set to 1, increases internal bias current by 50%.
LVDS_TXOPT[4:3]	For each bit: Default value is 0. When set to 1, decreases output current by 12.5%.
LVDS_TXOPT[2:1]	For each bit: Default value is 0. When set to 1, increases output current by 12.5%.

11.6.1.4 Unavailable features

- MCKO division factor of 3 is not supported on this device. Therefore, the setting PCR[MCKO_DIV] = 2 is not available
- Nexus DDR mode is not available on this device
- PSTAT mode is not available on this device

The following bits in the Port Configuration Register should always be written as zeros. These features are not available in the device:

- DDR_EN
- NEXCFG
- PSTAT_EN
- LP2_SYN

11.6.2 Block diagram

The following figure is a block diagram of the Nexus Port Controller (NPC) block.

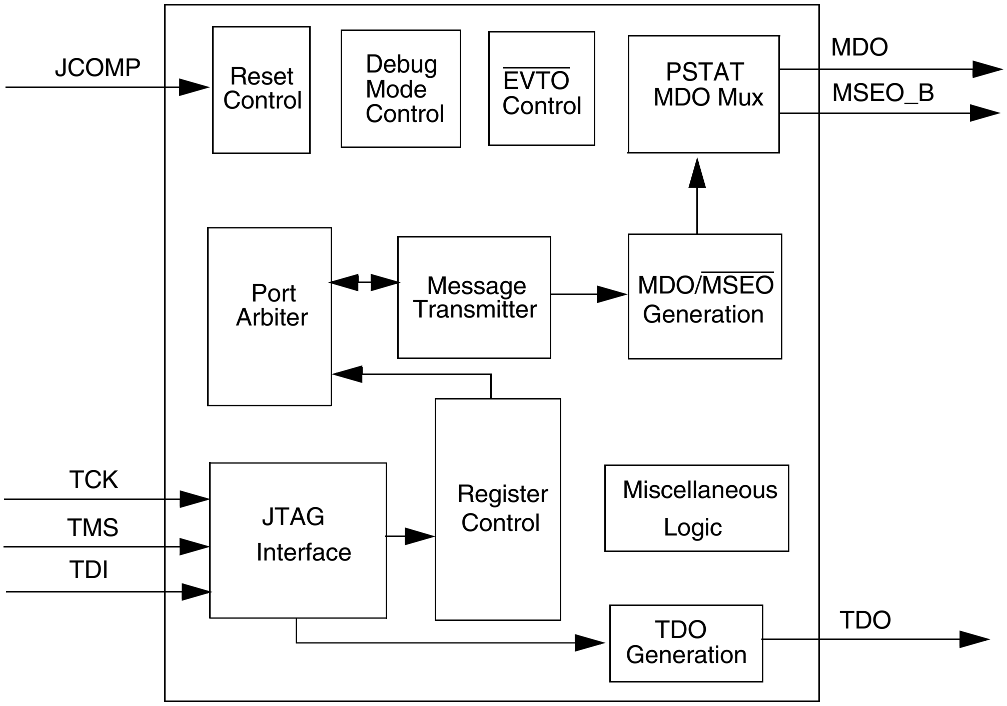


Figure 11-1. Nexus Port Controller Block Diagram

11.6.2.1 Overview

On a system-on-a-chip device, there are often multiple blocks that require development support. Each of these blocks implements a development interface based on the IEEE-ISTO 5001. The blocks share input and output ports that interface with the development tool. The NPC controls the usage of the input and output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

11.6.2.2 Features

The NPC block performs the following functions:

- Controls arbitration for ownership of the Nexus Auxiliary Output Port
- Nexus Device Identification Register and Messaging
- Generates MCKO enable and frequency division control signals
- Controls sharing of EVTO
- Generates an MCKO clock gating control signal to enable gating of MCKO when the auxiliary output port is idle
- Control of the device-wide debug mode
- Generates asynchronous reset signal for Nexus blocks based on JCOMP input, censorship status, and power-on reset status
- System clock locked status indication via MDO[0] following power-on reset
- Provides Nexus support for censorship mode
- Controls arbitration for ownership of the Nexus Aurora Link (NAL) in Aurora Link mode
- $\overline{\text{RDY}}$ pin support to increase the transfer rate of the IEEE 1149.1 port for large data read requests
- Low-power debug handshaking: see NPC Handshake

NOTE

MCKO may be gated one clock period early when MCKO's frequency is programmed as divide by 8 (MCKO_DIV= b111) and the MCKO gating function is enabled (MCKO_GT=1). In this case, the last MCKO received by the tool prior to the gating will correspond to the END_MESSAGE state. The tool will not receive an MCKO to indicate the transition to the IDLE state even though the NPC will transition to the IDLE state internally. Upon re-enabling of MCKO the first MCKO edge will drive the Message Start/End Output (MSEO=11) and move the tool's state to IDLE. Expect to receive the MCKO edge corresponding to the IDLE state upon re-enabling of MCKO after MCKO has been gated.

11.6.2.3 Modes of operation

The NPC block uses the JCOMP input, the censorship status, and an internal power-on reset indication as its primary reset signals. Upon exit of reset, the mode of operation is determined by the Port Configuration Register (PCR) settings. All available modes can be seen in Table 11-12.

11.6.2.3.1 Reset

The NPC block is asynchronously placed in reset when either power-on reset is asserted, JCOMP is not set for Nexus access, the device enters censored mode, or the JTAG Test Access Port (TAP) controller state machine is in the Test-Logic-Reset state. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. The NPC is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state
- The auxiliary output port pins are negated
- The TDI, TMS, and TCK TAP inputs are ignored (when in power-on reset, censored mode or JCOMP not set for NPC operation only)
- Registers default back to their reset values

11.6.2.3.2 Disabled-Port mode

In disabled-port mode, auxiliary output pin port enable signals are negated, thereby disabling message transmission. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through $\overline{\text{EVTO}}$.

11.6.2.3.3 Aurora Link mode (High Speed Debug mode)

In Aurora Link mode, trace data is transmitted through the Nexus Aurora Link (NAL) instead of the auxiliary port. For SoCs that support a Nexus Aurora Link, the FPM bit in the PCR functions as the Aurora Link enable, leaving reduced port mode (RPM) as the only available mode for data transmission via the auxiliary port. Aurora Link Mode is enabled by setting the MCKO_EN and FPM bits and selecting a non-zero value for the ALC_NUM_LN bits in the PCR. The MCKO_DIV value should be set to select a clock frequency equal to the system clock frequency (MCKO_DIV = 000).

11.6.2.3.4 Reduced-Port mode (Auxiliary Interface Debug mode)

Reduced-port mode (RPM) is entered by asserting the MCKO_EN bit and deasserting the FPM bit in the PCR. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific. Nexus Aurora Link is not available.

11.6.2.3.5 Censored mode

When the device is in censored mode, reading the contents of internal flash externally is not allowed. To prevent Nexus modules from violating censorship, the NPC is held in reset when in censored mode, asynchronously holding all other Nexus modules in reset as well. This prevents Nexus read/write to memory mapped resources and the transmission of Nexus trace messages.

11.6.3 External signal description

11.6.3.1 Overview

The NPC pin interface provides for the transmission of messages from Nexus blocks to the external development tools and for access to Nexus client registers. The following table outlines NPC pin definitions.

Table 11-9. NPC Signal Properties

Name	Port	Function	Reset State	Pull
$\overline{\text{EVTO}}$	Auxiliary	Event Out pin	0b1	-
$\overline{\text{EVTI\_IN}}$	Auxiliary	Event In pin	-	-
JCOMP	JTAG	JTAG Compliancy and TAP Sharing Control	-	Down
MDO[3:0]	Auxiliary	Message Data Out pins	0	-
$\overline{\text{MSEO}}$[1:0]	Auxiliary	Message Start/End Out pins	0b11	-
TCK	JTAG	Test Clock Input	-	Up
TDI	JTAG	Test Data Input	-	Up

Table continues on the next page...

Table 11-9. NPC Signal Properties (continued)

Name	Port	Function	Reset State	Pull
TDO	JTAG	Test Data Output	High Z	-
TMS	JTAG	Test Mode Select Input	-	Up
$\overline{\text{RDY}}$	JTAG	Data ready for transfer to/from Nexus Recommended Registers (NRRs; see also the IEEE-ISTO 5001-2003 standard)	-	-

11.6.3.2 Detailed signal descriptions

This section describes each of the signals listed in Table 11-9 in more detail. The JTAG test clock (TCK) input from the pin is not a direct input to the NPC. The NPC requires two separate input clocks for TCK clocked logic, one for posedge (rising edge TCK) logic and one for negedge (falling edge TCK) logic. Both clocks are derived from the pin TCK, and generated external to the NPC.

11.6.3.2.1 $\overline{\text{EVTO}}$- Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus blocks that implement the signal.

11.6.3.2.2 JCOMP - JTAG Compliancy

The JCOMP signal provides the ability to share the TAP. The NPC TAP controller is enabled when JCOMP is set to the NPC enable encoding; otherwise the NPC TAP controller remains in reset.

11.6.3.2.3 MDO - Message Data Out

Message Data Out (MDO) are output pins used for uploading Ownership Trace Messaging (OTM), Branch Trace Messaging (BTM), Data Trace Messaging (DTM) and other messages to the development tool. The development tool should sample MDO on the rising edge of MCKO. The width of the MDO bus used is determined by reset configuration.

11.6.3.2.4 MSEO_B - Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is an output pin that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool should sample $\overline{\text{MSEO}}$ on the rising edge of MCKO.

11.6.3.2.5 TCK - Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

The JTAG Clock (TCK) typically operates at a frequency well below the system clock frequency, as specified in the electrical data sheets. In some cases, however, such as low power mode, the system clock frequency may be lowered significantly from the normal operating range. If the system clock frequency is reduced below the frequency of TCK it will no longer be possible to communicate with the Nexus Port Controller Port Configuration Register (PCR). Therefore, if the tool needs to update the PCR Low Power Debug Enable (PCR[LP_DBG]) or Low Power Synchronization bits (PCR[LP1_SYN] and PCR[LP2_SYN]), the clock frequency must be lowered.

Ensure that the frequency of TCK does not exceed the system clock frequency during normal operation and during low power operation. For usage of the NPC/Nexus/NXSS subsystem, TCK clock frequency needs to be smaller than SYSCLK/2.

11.6.3.2.6 TDI - Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

11.6.3.2.7 TDO - Nexus Test Data Output

Test Data Output (TDO) pin transmits serial output for instructions and data. TDO is three-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

11.6.3.2.8 TMS - Test Mode Select

Test Mode Select Input (TMS) pin is used to sequence the IEEE1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

11.6.3.2.9 $\overline{\text{RDY}}$ - Data ready for transfer

The $\overline{\text{RDY}}$ pin exists to increase the transfer rate of the IEEE 1149.1 port. It is used to signal when data are ready to be transferred to and from the NRRs (see also IEEE-ISTO 5001-2003). This may eliminate the need to poll NRRs for status information for synchronization purposes. This capability becomes especially important when performing read/write access transfers to different speed target memories.

The $\overline{\text{RDY}}$ pin asserts (asynchronously) a logic low whenever the read/write access transfer has completed without error and then deasserts when the IEEE 1149.1 state machine has reached the CAPTURE_DR state.

The $\overline{\text{RDY}}$ pin is controlled by the TEST_CTRL register in the JTAGC module. See the JTAG Controller (JTAGC) chapter.

11.6.4 Register definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

The following table shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC block does not implement the client select control register because the value does not matter when accessing the registers. Note that the bypass and instruction registers have no index values. These registers are not accessed in the same manner as Nexus client registers. Refer to the individual register descriptions in Nexus Port Controller (NPC) chapter for more details.

Table 11-10. NPC Registers

Index	Register
0	Device ID Register (DID)
126	Aurora Configuration Register (ACR)
127	Port Configuration Register (PCR)

11.6.4.1 Register descriptions

11.6.4.1.1 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

11.6.4.1.2 Instruction Register

The NPC block uses a 4-bit instruction register as shown in the following table. The instruction register is accessed via the SELECT_IR_SCAN path of the TAP controller state machine, and allows instructions to be loaded into the block to enable the NPC for register access (NEXUS_ENABLE; see also Table 11-21) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

NOTE

For details, see Figure 11-5.

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in synchronous loading of the BYPASS instruction. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	0	1	2	3
R	Previous Instruction Opcode			
W	Instruction Opcode			
Reset:	BYPASS Instruction Opcode (0xF)			

11.6.4.1.3 Nexus Device ID Register (DID)

The device identification register allows the part revision number, design center, part identification number, and manufacturer identity code of the part to be determined through the auxiliary output port.

11.6.4.1.4 Aurora Configuration Register (ACR)

The ACR is used to configure the RXOPT and TXOPT settings of the LVDS receive and transmit pads.

11.6.4.1.5 Port Configuration Register (PCR)

The PCR is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

The PCR register may be rewritten by the debug tool subsequent to the enabling of the NPC for low power debug support. In this case, the debug tool may set and clear the LP_DBG and LPn_SYN bits, but must preserve the original state of the remaining bits in the register. For details, see NPC Handshake.

NOTE

> The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

11.6.5 Functional Description

11.6.5.1 NPC reset configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the auxiliary port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

For Nexus Aurora Link, asserting the MCKO_EN and FPM bits and setting the NUM_ALC_LN bits to a non-zero value enables NAL data transmission. Transmission of data via the auxiliary port is enabled only by selecting the RPM configuration in this case.

The following table describes the NPC reset configuration options.

Table 11-12. NPC Reset Configuration Options

JCOMP Equal to 1?	MCKO_EN bit of the Port Configuration Register	NUM_ALC_LN bits of the Port Configuration Register > 0?	FPM bit of the Port Configuration Register	Configuration
no	X	X	X	Reset
yes	0	X	X	Disabled
yes	1	yes	1	Aurora Link Mode
yes	1	X	0	Reduced-Port Mode

NOTE

> If the device is censored, the NPC is held in reset, asynchronously holding all other Nexus modules in reset as well

11.6.5.2 Auxiliary output port

The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the Nexus modules and arbitrates for access to the port.

11.6.5.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the $\overline{\text{MSEO}}$ functions. The $\overline{\text{MSEO}}$ pins are used to signal the end of variable-length packets and the end of messages. They are not required to indicate the end of fixed-length packets. MDO and $\overline{\text{MSEO}}$ are sampled on the rising edge of MCKO.

The following figure illustrates the state diagram for $\overline{\text{MSEO}}$ transfers. All transitions not included in the figure are reserved, and must not be used.

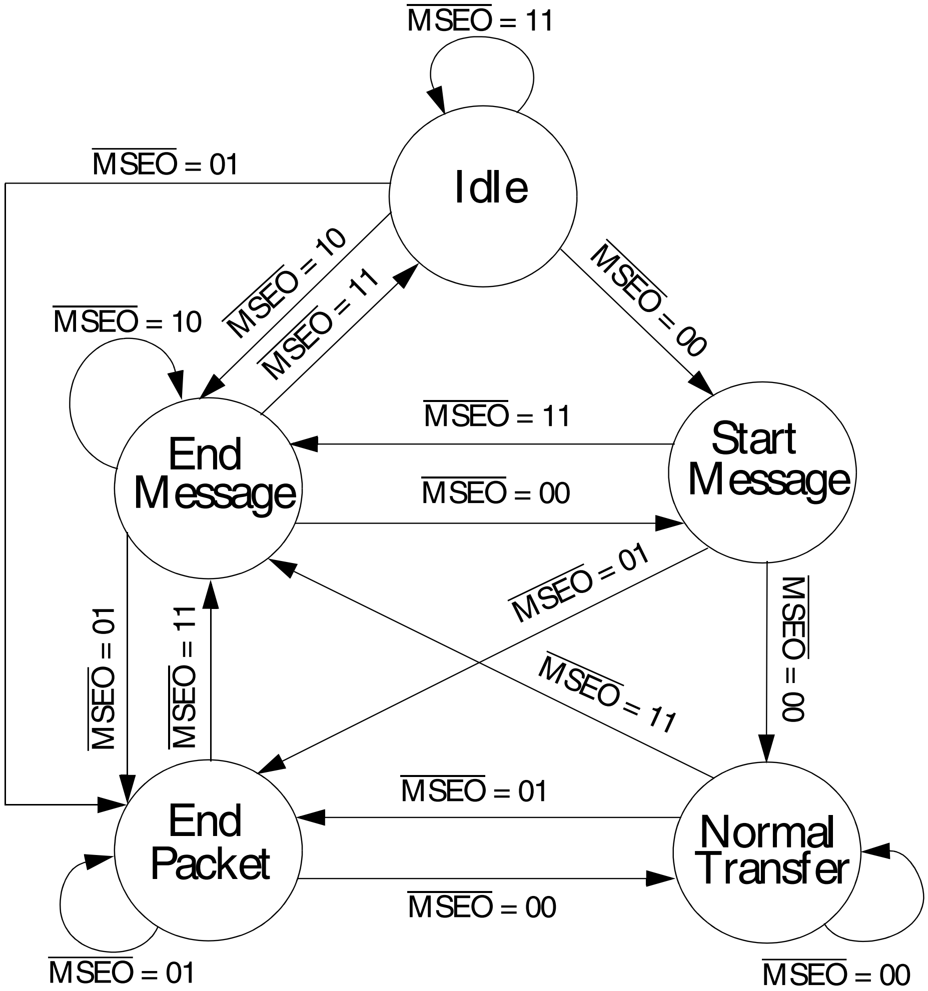

$\overline{\text{MSEO}}$ = 11
Idle
$\overline{\text{MSEO}}$ = 01
$\overline{\text{MSEO}}$ = 10
$\overline{\text{MSEO}}$ = 11
$\overline{\text{MSEO}}$ = 00
$\overline{\text{MSEO}}$ = 10
End Message
$\overline{\text{MSEO}}$ = 11
Start Message
$\overline{\text{MSEO}}$ = 00
$\overline{\text{MSEO}}$ = 01
$\overline{\text{MSEO}}$ = 11
$\overline{\text{MSEO}}$ = 01
$\overline{\text{MSEO}}$ = 11
$\overline{\text{MSEO}}$ = 00
End Packet
$\overline{\text{MSEO}}$ = 01
Normal Transfer
$\overline{\text{MSEO}}$ = 00
$\overline{\text{MSEO}}$ = 01
$\overline{\text{MSEO}}$ = 00

Figure 11-2. $\overline{\text{MSEO}}$ bit transfers (for 2-bit $\overline{\text{MSEO}}$)

11.6.5.2.2 Output Messages

In addition to sending out messages generated in other Nexus blocks, the NPC can also output the device ID message contained in the device ID register and the port replacement output message on the MDO pins. The device ID message can also be sent out serially through TDO.

The following table describes the device ID and port replacement output messages that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Table 11-13. NPC Output Messages

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Device ID Message	6	6	fixed	TCODE	Value = 1

Table continues on the next page...

Table 11-13. NPC Output Messages (continued)

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
	32	32	fixed	ID	DID register contents
Aurora Link Overrun Error Message	6	6	fixed	TCODE	Value = 8
	4	4	fixed	SRC	Nexus Aurora Link SRC value (SRC=0xF).
	5	5	fixed	ECODE	Value = 01000. Error code indicating overrun of NAL FIFO.

The following table shows the various message formats that the pin interface formatter must encounter. Note that for variable-length fields, the transmitted size of the field is determined from the range of the least significant bit to the most significant non-zero-valued bit (i.e. most significant zero-valued bits are not transmitted).

Table 11-14. Message Field Sizes

Message	TCODE	Field #1	Field #2	Field #3	Field #4	Field #5	Min. Size[1] (bits)	Max Size[2] (bits)
Device ID Message	1	Fixed = 32	NA	NA	NA	NA	38	38
Aurora Link Overrun Error Message	8	Fixed=4	Fixed=5	N/A	N/A	N/A	15	15

[1] Minimum information size. The actual number of bits transmitted depends on the number of MDO pins.

[2] Maximum information size. The actual number of bits transmitted depends on the number of MDO pins.

The double edges in the preceding table indicate the starts and ends of messages. Fields without shaded areas between them are grouped into super-fields and can be transmitted together without end-of-packet indications between them.

11.6.5.2.3 Rules of Message

- A variable-sized field within a message must end on a port boundary. (Port boundaries depend on the number of MDO pins active with the current reset configuration.)
- A variable-sized field may start within a port boundary only when following a fixed-length field.
- Super-fields must end on a port boundary.

- When a variable-length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary.
- Multiple fixed-length packets may start and/or end on a single clock.
- When any packet follows a variable-length packet, it must start on a port boundary.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information.
- Within a field, the lowest significant bits are shifted out first. Figure 17 shows the transmission sequence of a message that is made up of a TCODE followed by two fields.

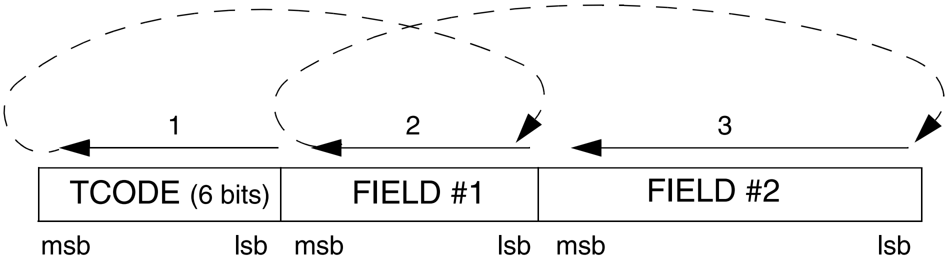


Figure 11-3. Transmission sequence of messages

11.6.5.3 IEEE 1149.1-2001 (JTAG) TAP

The NPC block uses the IEEE 1149.1-2001 TAP for accessing registers. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. There may also be other blocks on the MCU that use the TAP and implement a TAP controller. The value of the JCOMP input controls ownership of the port between Nexus and non-Nexus blocks sharing the TAP.

Refer to the IEEE 1149.1-2001 specification for further detail on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1-2001 state machine shown in Figure 11-5. The Nexus controller state machine is defined by the IEEE-ISTO 5001-2003 standard. It is shown in Figure 11-6.

The instructions implemented by the NPC TAP controller are listed in the following table. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4-bits.

Table 11-15. Implemented Instructions

Instruction Name	Private/Public	Opcode	Description
NEXUS-ENABLE	public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

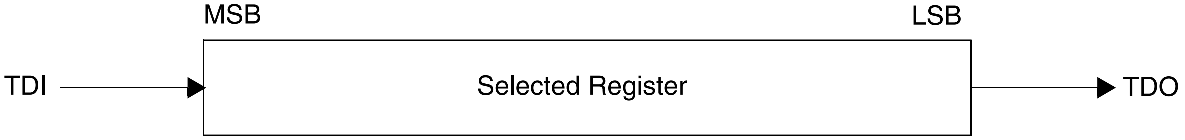
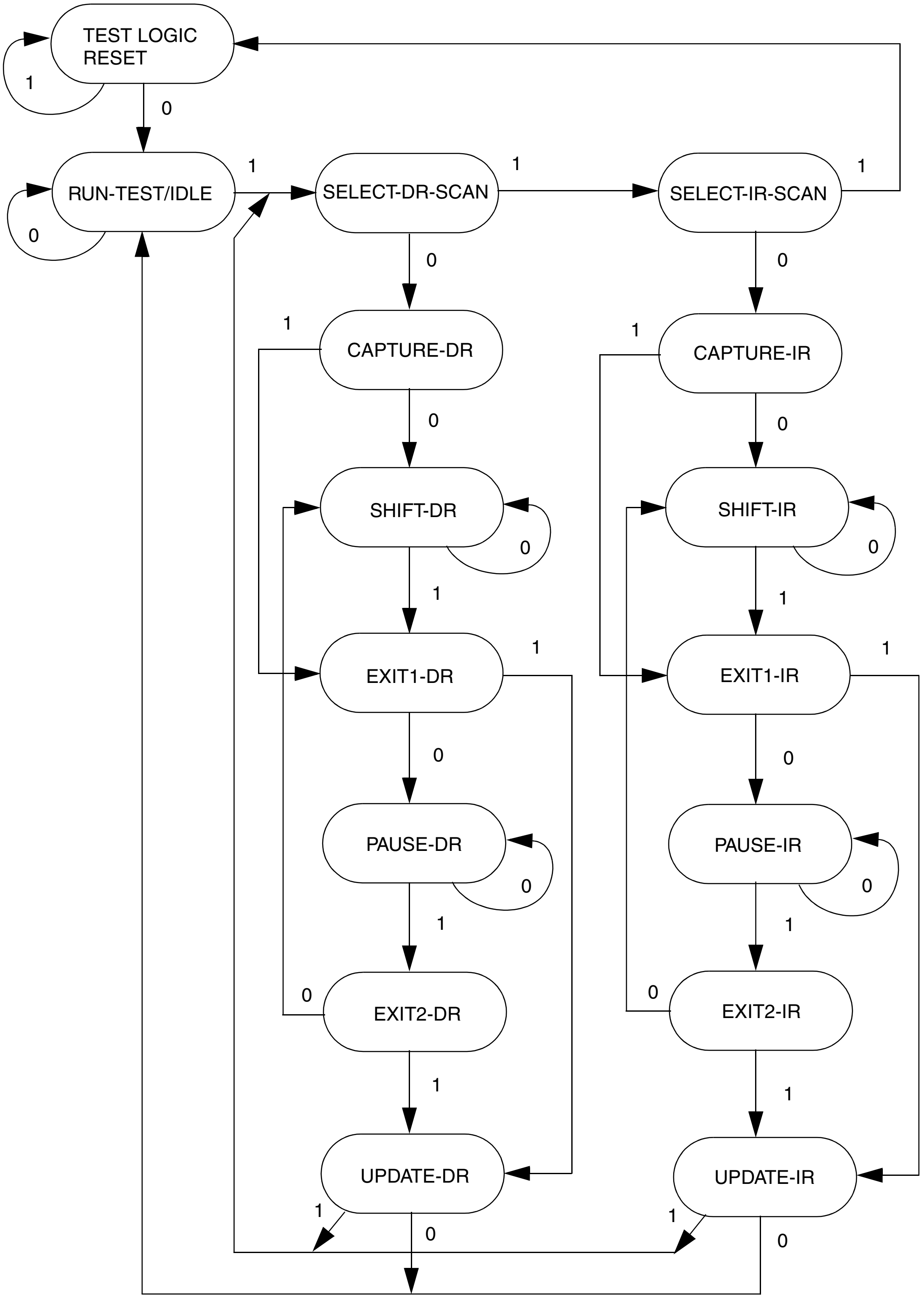


Figure 11-4. Shifting data into register

11.6.5.3.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal, entry into censored mode, or setting JCOMP to a value other than the NPC enable encoding resets the NPC TAP controller. When not in power-on reset or censored mode, the NPC TAP controller is enabled by driving JCOMP with the NPC enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 11-5. IEEE 1149.1-2001 TAP Controller state machine

11.6.5.3.2 Retrieving device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the auxiliary output port or shifted out serially by accessing the Nexus Device ID register through the TAP. Transmission of the device identification message on the auxiliary output port MDO pins occurs immediately after a write to the PCR, if the NPC is enabled. Transmission of the device identification message serially via TDO is achieved by performing a read of the register contents as described in "Selecting a Nexus Client Register" section .

11.6.5.3.3 Loading NEXUS-ENABLE Instruction

Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 20 transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. Following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

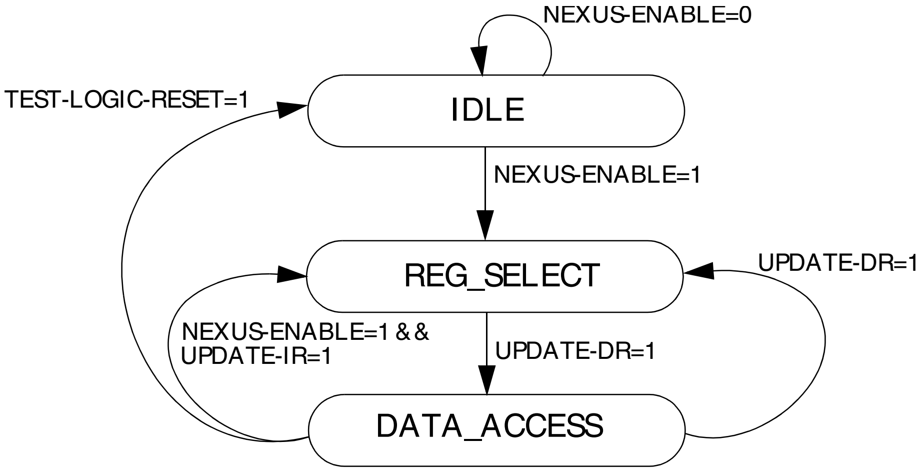


Figure 11-6. NEXUS controller state machine

Table 11-16. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction

Table continues on the next page...

Table 11-16. Loading NEXUS-ENABLE instruction (continued)

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
3 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

11.6.5.3.4 Selecting a Nexus Client register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in Figure 21. The read/write control bit is set to 1 for writes and 0 for reads.

Table 11-17. IEEE 1149.1 Controller Command Input

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE

1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

Following table illustrates a sequence which writes a 32-bit value to a register

Table 11-18. Write to a 32-Bit Nexus Client register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
48	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
49	1	UPDATE-DR	DATA_ACCESS	Value written to register
50	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

11.6.5.4 Nexus JTAG port sharing

Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register.

11.6.5.5 MCKO

MCKO is an output clock to the development tools used for the timing of MSEO and MDO pin functions. MCKO is derived from the system clock and its frequency is determined by the value of the MCKO_DIV field in the PCR. Possible operating frequencies include system clock, one-half system clock, one-quarter system clock, and one-eighth system clock speed.

The NPC also generates an MCKO clock gating control output signal. This output can be used by the MCKO generation logic to gate the transmission of MCKO when the auxiliary port is enabled but not transmitting messages. The setting of the MCKO_GT bit inside the PCR determines whether or not MCKO gating control is active. The MCKO_GT bit resets to a logic 0. In this state gating of MCKO is disabled. To enable gating of MCKO, the MCKO_GT bit in the PCR is written to a logic 1.

11.6.5.6 $\overline{\text{EVTO}}$ Sharing

The NPC block controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. The NPC assumes incoming $\overline{\text{EVTO}}$ signals will be asserted for one system clock period. After receiving a single clock period of asserted $\overline{\text{EVTO}}$ from any Nexus client, the NPC latches the result, and drives $\overline{\text{EVTO}}$ for one MCKO period on the following clock. When there is no active MCKO, such as in disabled mode, the NPC drives $\overline{\text{EVTO}}$ for two system clock periods. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

11.6.5.7 Nexus Reset Control

The JCOMP input that is used as the primary reset signal for the NPC is also used by the NPC to generate a single-bit reset signal for other Nexus blocks. If JCOMP is negated, an internal reset is asserted, indicating that all Nexus modules should be held in reset. Internal Nexus reset is also asserted when the device is in censored mode.

11.6.5.8 System clock locked indication

Following a power-on reset, MDO[0] can be monitored to provide the lock status of the system clock. MDO[0] is driven to a logic 1 until the system clock achieves lock after exiting power-on reset. Once the system clock is locked, MDO[0] is negated and tools may begin Nexus configuration. Loss of lock conditions that occur subsequent to the exit of power-on reset and the initial lock of the system clock do not cause a Nexus reset, and therefore do not result in MDO[0] driven high.

11.6.5.9 Data Transmission via Nexus Aurora Link

As an alternative to data transmission on the Nexus auxiliary port, the NPC also supports data transmission over a high-speed serial Nexus Aurora Link (NAL). The NPC's role in Aurora Link data transmission is to route trace data to the NAL while continuing to manage arbitration between Nexus clients for the right to transmit data. In addition, the NPC monitors the NAL transmit queue and will halt further data transmission from Nexus clients if the queue is in danger of an overrun condition. In the event of an overrun condition, the NPC transmits a standard Nexus error message.

Once the PCR is configured for Aurora Link Mode (MCKO_EN bit set, FPM bit set, ALC_NUM_LN set to non-zero value), the NPC waits for the NAL to acknowledge it is ready before starting data transmission. The NPC IDCODE message is the first message transmitted after Aurora Link mode is successfully configured and the NAL acknowledges it is ready to receive data. All data transmitted in Aurora Link mode is transmitted in frames, with each frame consisting of 8192 bytes of data. The first beat of each frame is accompanied by the assertion of the start of frame control signal, and the last beat of each frame is accompanied by the assertion of the end of frame control signal. Each beat of valid NAL data from the NPC is also accompanied by the assertion of the NAL data valid message signal.

Data received by the NAL is held in a FIFO until it can be sent out on the Aurora Link. The NPC keeps track of the NAL FIFO fill level. To help prevent an overrun of the NAL FIFO, the NPC holds off further grants to all clients if the NAL FIFO is filled above the watermark level. In the event that the NAL FIFO is overrun with data, the NPC transmits its FIFO overrun error message shown in Table 11-13.

11.6.6 Initialization/Application Information

11.6.6.1 Accessing NPC tool-mapped registers

To initialize the TAP for Nexus register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Load the TAP controller with the NEXUS-ENABLE instruction

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.
2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.
2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE-ISTO 5001 standard for more detail.

11.7 NPC Handshake

11.7.1 Overview

The NPC_HNDSHK manages the low-power mode debug handshaking between the Nexus Port Controller (NPC) and the mode entry module (MC_ME).

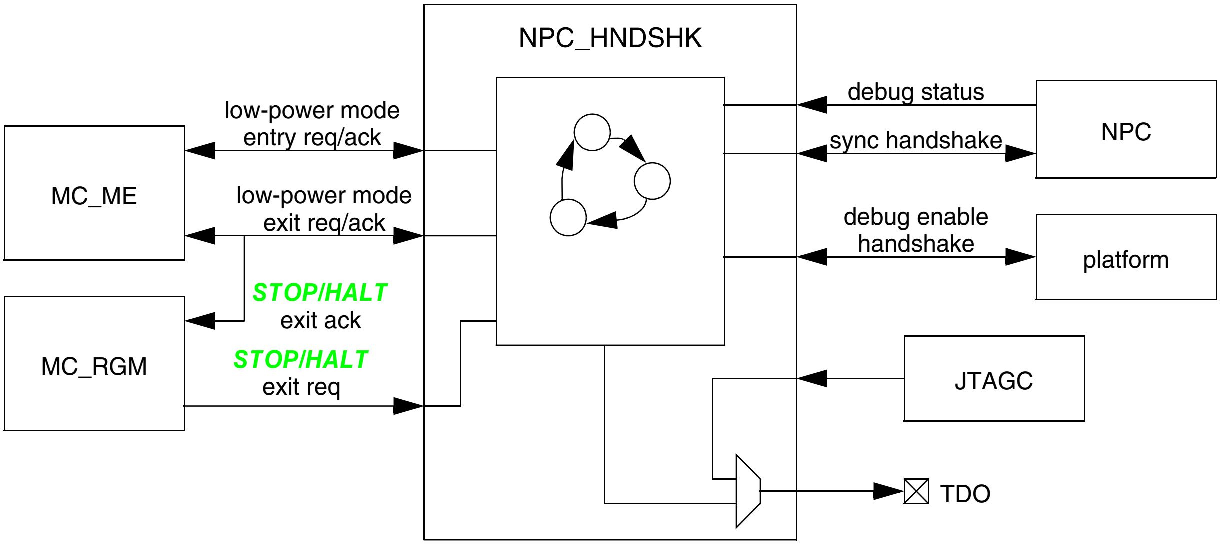


Figure 11-7. NPC_HNDSHK block diagram

11.7.2 Features

The NPC_HNDSHK supports:

- setting and clearing of the NPC PCR sync bit on low-power mode entry and exit
- putting the core into debug mode on low-power mode exit
- generating a falling edge on the JTAG TDO pad on low-power mode exit

11.7.3 Functional Description

11.7.3.1 Operation

The NPC_HNDSHK implements the state machine shown in following figure. The states are described in the following table.

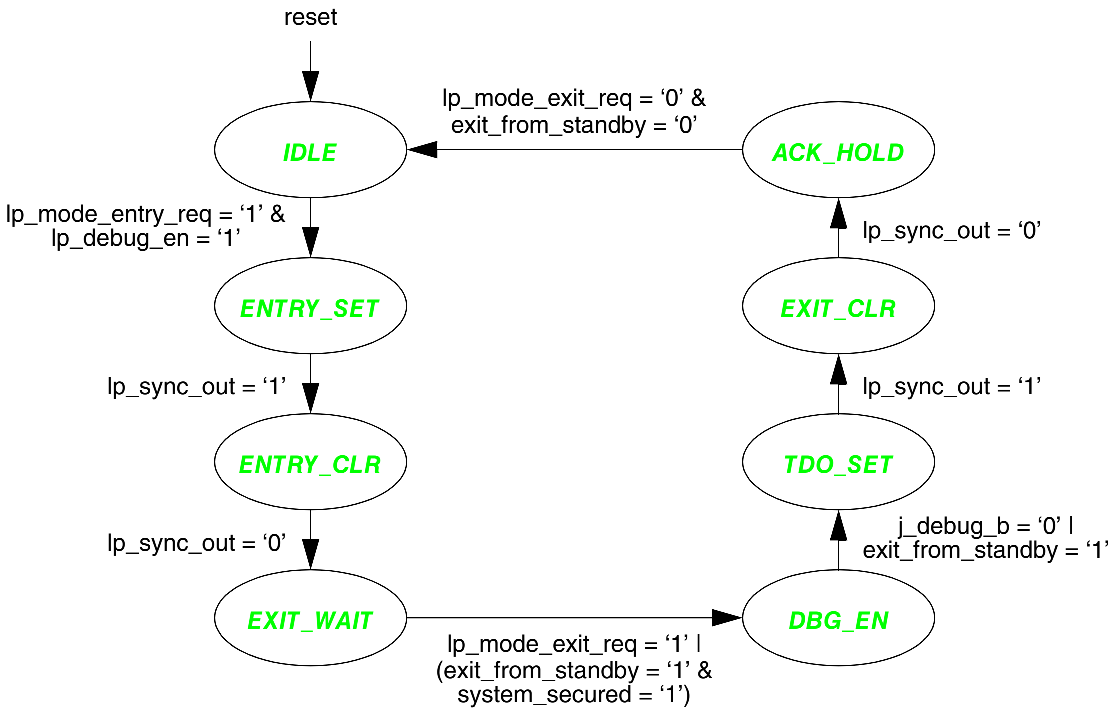


Figure 11-8. NPC_HNDSHK state machine

Table 11-19. NPC_HNDSHK state coding

Value	State	Description
000	IDLE	wait for low-power mode entry request assertion
001	ENTRY_SET	set NPC PCR sync bit
011	ENTRY_CLR	wait for NPC PCR sync bit clear
010	EXIT_WAIT	assert JTAG TDO pad and wait for low-power mode exit request assertion
110	DBG_EN	enable core debug mode
111	TDO_SET	deassert JTAG TDO pad and wait for NPC PCR bit set
101	EXIT_CLR	clear NPC PCR sync bit and return TDO pad to normal functionality
100	ACK_HOLD	wait for low-power mode exit request deassertion

11.8 Nexus Aurora interface

The Aurora interface allows Nexus protocol information to be transmitted serially at a high rate of speed over multiple Aurora lanes to provide advanced trace information from the device. The Aurora protocol handles the encoding of the data and striping the information across the lanes.

The Aurora information can be used to reconstruct events or operations that occurred inside the microcontroller. Nexus supports multiple trace clients within the MCU to all transmit information through a single Aurora port. Each Nexus message is tagged with the source client identifier and the type of message. The actual trace information that is available depends on the Nexus client and, generally, can be individually enabled (both the type of messages and which clients). The following table shows the different types of Nexus messages.

Table 11-20. Types of Nexus messages

Message Type	Description
Program Trace	Nexus Program trace messages transmit any discontinuities in the program flow, such as branches and interrupts. Multiple types of messages can be generated.
Ownership Trace	Ownership Trace provides information on process identification changes.
Watchpoint Trace	Watchpoint messages are generated anytime a watchpoint match occurs in the program or data flow. These can be programmed for many types of events within the MCU.
Device Identification	The device identification message allows information about the MCU to be transmitted on power up to allow tools to identify the target system MCU type.
Debug Status	Debug status message provides additional information that may be needed to reconstruct software execution. Information such as if the device has entered debug mode or low power mode.
Data Acquisition	Data Acquisition messages are an optional feature that allows software control of information to be transmitted.
Error	Error messages are transmitted when an error condition occurs, such as internal buffer overflows.

The Nexus trace information can be output to internal memory or the parallel information can be serialized for transmission over a serial type interface through the Nexus Aurora interface.

11.8.1 Nexus Aurora overview

The IEEE-ISTO 5001 standard added a new type of Auxiliary trace port that supports a higher bandwidth. This new interface type uses the Aurora protocol to reformat the parallel Nexus messages into one or more output serial lanes of data, where a lane is defined as a 2-pin Low Voltage Differential Signals (LVDS) interface. The protocol handles striping the data onto the multiple lanes. Each lane uses an 8B/10B encoded LVDS drivers. The encoding allows the receiver to recover clocking information.

The Aurora physical interface is based on XAUI (10 Gigabit Attachment Unit Interface used in communication protocols) and covers the jitter, differential skew, inter-lane skew, and bit error rate requirements.

The Aurora protocol covers lane initialization and channel bonding for both simplex and duplex implementations. The current implementation implements a simplex output only connection from the MCU to the external tool.

The Aurora interface connects Nexus parallel trace output to an Aurora Protocol Engine (APE), the Aurora Lane Control (ALC) and the Nexus Aurora Physical interface (NAP). The following figure is the Aurora interface block diagram, where the number of lanes, n, is 2.

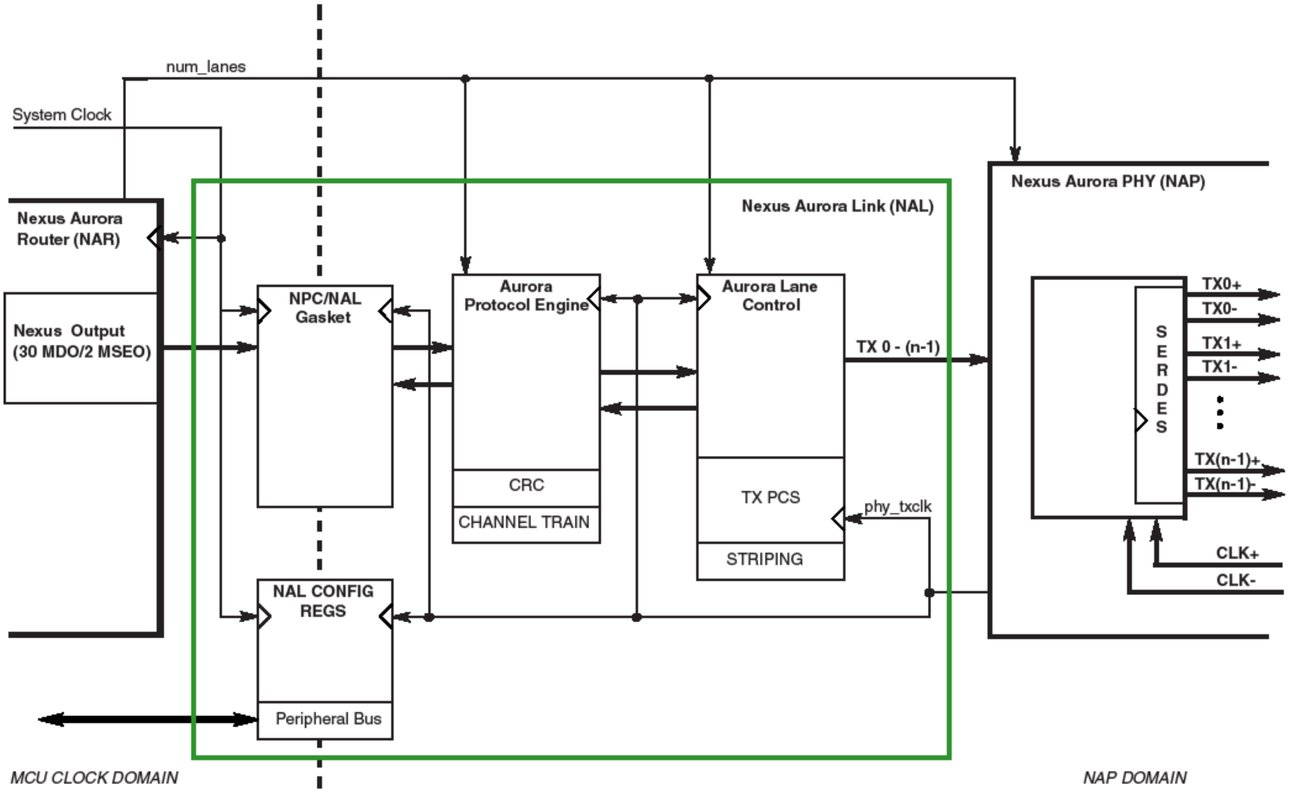


Figure 11-9. Aurora interface block diagram

The Nexus blocks do not have memory mapped registers. The Nexus registers are accessed by the development tool via the JTAG port using a two step process. First, the specific block is selected by loading the corresponding ACCESS_AUX instruction as described in JTAGC ACCESS_AUX block instructions. Next, after the block is selected, the Nexus registers are enabled by loading the enable instruction to the JTAGC Instruction Register (IR). The enable instructions for the different Nexus clients are shown in the following table. After the enable instruction is received, the development tool has access to the Nexus registers of the selected client.

Table 11-21. Nexus client JTAG enable instructions

Module	Enable instruction	Opcode
Cores (e200z4)	CORE_ENABLE	0x7C (10 bits)
NXMC modules	NEXUS_ENABLE	0x0 (4 bits)
NPC	NEXUS_ENABLE	0x0 (4 bits)
NAL	NEXUS_ENABLE	0xE (4 bits)

11.8.2 Nexus Aurora clocking

The supported clock frequency on the Aurora clock input has a valid range of 625 MHz to 1.25 GHz.

The Aurora interface requires the NAL and NAP to run with the same clock frequency as the cores for maximum bandwidth. For a 1.25 GBit/s data rate, the core frequency must be >=125 MHz.

11.9 Nexus Aurora Link (NAL)

The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer serializer and transmitter (no receiver). It includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol.

Debug data in the Nexus format from multiple clients is packetized into 32-bit format by the clients and then collected by the NPC. The NAL encodes the message data (8/10), stripes then across the lanes, and inserts Aurora control characters (such as start of message, end of message, idle, clock compensation characters and so on) around the payload. The NAL PCS drives the Nexus Aurora Physical (NAP) interface.

11.9.1 Chip-specific NAL information

Information about the NAL module on this chip includes the details in the following subsections.

11.9.1.1 Features

- Xilinx Aurora Protocol Specification V2.0 compliant.
- Two transmit-only lanes.
- Max data transmit rate = 2.5Gb/s with 2 transmit lanes running at 1.25 GHz.
- Cyclic Redundancy Checking (CRC) for error detection.
- Static (timer-based) channel training.
- Native and User Flow Control support.

11.9.1.2 NAL_GSR reset value

On this chip, the reset value of NAL_GSR is 0000_2400h.

See Nexus Aurora Link General Status Register (NAL_GSR) for details about this register.

11.9.2 Modes of Operation

The operating mode of the Aurora Link is determined first by the main SOC control signals (reset, scan_en). Its functional configuration is determined by the main SOC IO Port Control setting; functional operation is enabled by the PCR register in the NPC.

11.9.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Note that, since there is no tck-domain logic in the link itself, it is unaffected by treset.

Depending on the configuration, the link may power up upon exiting reset mode, if any Aurora lanes are enabled during reset.

11.9.2.2 Power-down

During purely functional operation, Nexus trace messaging is inhibited and clocks are disabled to the blocks of the Aurora Link. If the SOC exits POR-config with the Aurora channel disabled, the Aurora Link and any SerDes lanes dedicated to Aurora remain powered down. However, if the Link is powered down but the NPC is subsequently enabled with the Aurora channel active, the Link and dedicated SerDes lanes power up at this time.

11.9.2.3 Link Training

Before beginning functional operation, the Aurora Link must go through a training process to insure that both sides (SOC and debugger) can correctly reconstruct the transmitted data stream. The training process is outlined in the Aurora Protocol Specification, and consists of three sections: lane initialization, channel bonding, and channel verification. Training proceeds once the link powers up, and must complete successfully before data transmission can occur.

The NAL can be configured to initialize and train the Aurora link only in a simplex configuration. In training mode a timer based static-training method is used. The initialization and training sequence progresses from one step to the next via the use of programmable timers. A timer is programmed for each stage of training and as each timer expires, the training sequence can proceed to the next stage.

11.9.2.4 Aurora Transmit

During normal operation, the NPC will push data into the Aurora Link at a sufficiently high rate to insure that the link will not become data-starved even in its highest-bandwidth configuration (2 lanes operating at 1.25 GHz). The Link then performs the buffering, domain-crossing, data-packaging, and protocol translations to prepare the data stream for transmission through the NAP block.

11.9.3 Memory Map and Register Description

Configure the Aurora Link by programming the PCR register in the NPC and registers inside the NAL via JTAG accesses. Additional configuration registers within the link itself pertain to static training and debug modes. The link also has a read-only status register that provide information on channel training and general link operation.

- Nexus Aurora Link Status Register (NALGSR): The NALGSR register reflects general status of the Aurora link.
- Nexus Aurora Link Error Status Register (NALESR): The NALESR register reports error status in the Aurora link.
- NAL General Control Register (NALGCR): The NALGCR configures the behavior of the APE. It can be programmed to reset the Aurora channel, or to configure simplex/duplex initialization and to enable debug features.
- NAL Training Control Register (NALTCR): The NALTCR configures static training and timeout controls for the APE.

For more details about the register definition, see the Nexus Aurora Link (NAL) chapter.

11.10 Nexus Aurora PHY (NAP)

The Nexus Aurora PHY (NAP), in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. The NAP receives 8b/10b encoded data from the ALC block in the NAL and transmits this data onto a given PHY Lane's LVDS pads. The NAL performs lane striping with settings in the NAL and NPC specifying lane configuration. Each NAP lane consists of a Data Symbol Character Accumulator which registers the 10-bit character for that lane from the ALC on the rising edge of the NAL clock. A Serializer takes the 10-bit character and serializes it into the individual transmit data bits at the PHY clock rate to be transmitted via the LVDS output drivers to the external debugger receive channel.

NOTE

> The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive data side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called Lane Alignment in Aurora terminology), 10b/8b decoding, decode and disparity error tracking, and CRC checking in the received data stream.

The following figure shows a block diagram of the Nexus Aurora Physical (NAP) interface and the LVDS output drivers, where the number of lanes, n, is 2.

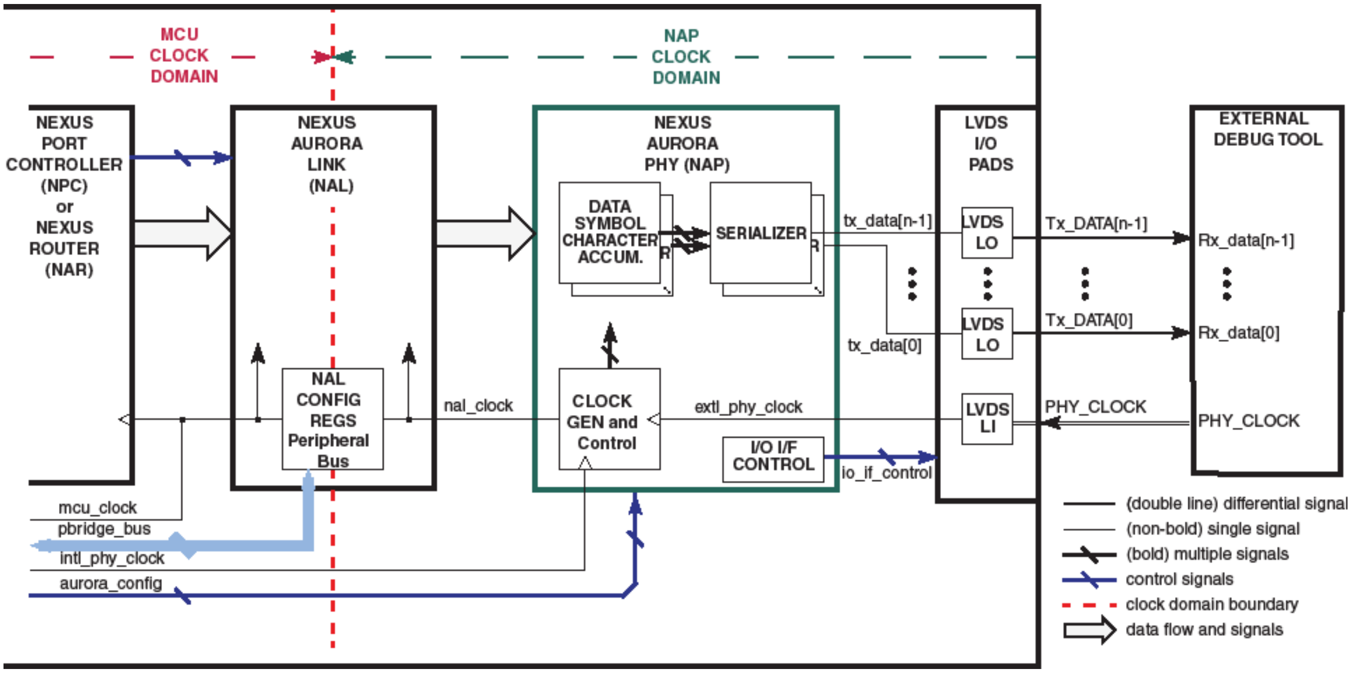


Figure 11-10. NAP block diagram

11.10.1 Features

The features of the NAP are as follows:

- Xilinx Aurora Protocol Specification V2.0 compliant
- Two differential transmit-only lanes
- Max data transmit rate of 2.5 Gb/s with two transmit lanes running at 1.25 GHz
- Simplex mode with static (timer-based) channel training

11.10.2 Modes of operation

The operating mode of the NAP is determined by the SoC control signals. The functional configuration is determined by the NPC PCR and NAL internal register settings; functional operation is enabled by the NPC.

11.10.2.1 Reset

During Nexus Reset all internal NAP resources are reset; all output ports are disabled, and all inputs are ignored. Since there is no SoC clock domain logic in the NAP itself, the NAP is unaffected by SoC system reset. Depending on the NAP configuration settings, the NAP may activate upon exiting Nexus Reset if any Aurora lanes are enabled during this reset mode, else it remains disabled and inactive.

11.10.2.2 Disabled/Enabled

The NAP is disabled when it comes out of reset if the SoC is configured for pure SoC functional mode operation with Nexus trace messaging inhibited. The Aurora LVDS transmit lanes also remain powered down. In addition, any time the NAP block enable is negated the NAP is disabled and the LVDS transmit lanes powered down. Conversely, if the NAP is enabled when it comes out of reset or enabled sometime later after reset is negated, the NAP channels become active and the LVDS output lanes are powered on to transmit Nexus trace data.

11.10.2.3 Link training

Before beginning trace output mode operation, the Aurora Link must go through a training process conducted by the NAL to insure that an external debugger can correctly reconstruct the transmitted data stream. Training proceeds once the link powers up, and must complete successfully before data transmission can occur.

The NAL initializes and trains the Aurora Link in a simplex configuration using a timer based static-training method. The initialization and training sequence described in Aurora Protocol Specification progresses from one step to the next via the use of programmable timers contained in the NAL. A timer is programmed for each stage of training and as each timer expires, the training sequence proceeds to the next stage until complete. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

11.10.2.4 Transmit

During Nexus trace output operation, it is the responsibility of the NPC to push data into the Aurora Link at a sufficiently high rate to insure that the link does not become data-starved even in its highest-bandwidth configuration. The NAL and NAP support only two Tx lane configuration. The number of Tx lanes available is configured at reset.

11.10.3 Signals

The following table correlates module-level signal names (shown in the NAP's dedicated chapter) and the corresponding chip-level names (shown in the Signal Description chapter).

Table 11-22. NAP signals

Module signal name	Chip signal name	Direction	Description
PHY_CLOCK	AUR_RXCLK_N	Input	Aurora Receive clock, Negative differential pair (LVDS)
	AUR_RXCLK_P	Input	Aurora Receive clock, Positive differential pair (LVDS)
TX Data0	AUR_TX0_N	Output	Aurora Transmit lane 0, Negative differential pair (LVDS)
	AUR_TX0_P	Output	Aurora Transmit lane 0, Positive differential pair (LVDS)
TX Data1	AUR_TX1_N	Output	Aurora Transmit lane 1, Negative differential pair (LVDS)
	AUR_TX1_P	Output	Aurora Transmit lane 1, Positive differential pair (LVDS)

11.11 Nexus clients

Nexus supports multiple trace clients within the device to all transmit information through a single Aurora port.

Table 11-23. Reference links to related information

Related module	Reference
e200z4 Nexus Support	e200z4 Nexus 3 Module
Nexus Crossbar Multi-Master Client (NXMC)	Nexus Crossbar Multi-master Client (NXMC)

The following table lists all possible clients that can create trace messages in the device and the SRC IDs used.

Table 11-24. Trace message SRC IDs used

Client	SRC ID used in trace messages
NPC (Aurora link overrun error message)	0xF
e200z4 Nexus	0x0
Nexus Crossbar Multi-Master Client 0 (DMA/SIPI/Ethernet)	0xC
Nexus Crossbar Multi-Master Client 1 (FlexRay)	0xD

11.11.1 e200z4 Nexus 3

The Nexus3 module provides real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

11.11.2 Nexus Crossbar Multi-master Client (NXMC)

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities for SoCs without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE1149.1-2001 JTAG standard.

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) is implemented that monitors traffic into the master port of the XBAR. Two NXMCs are implemented, for the M5 and M6 master ports at the XBAR. Trace messages transmitted over the Nexus interface include which NXMC generated the message, as well as an identifier of the pre-concentrator source of the message.

This device has two available instances of the NXMC module:

- NXMC0 is attached to AHB master port M5 and traces accesses to DMA/SIPI/ Ethernet peripherals; the message SRC ID is Ch (see also Table 11-24)
- NXMC1 is attached to AHB master port M6 and traces accesses to the FlexRay; the message SRC ID is Dh (see also Table 11-24)

11.11.2.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001 standard. The following features are supported:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources.
- Multi-master tracing capability for multiple master accesses
- Watchpoint trigger enable of data trace messaging
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

11.11.2.2 External signal description

This section details information regarding the standard ports and protocol. The NXMCpin interface provides the function of transmitting messages from the messages queues to the NPC. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$ (Event In)
- One $\overline{\text{EVTO}}$ (Watchpoint Event Out)
- Two$\overline{\text{MSEO}}$ (Message Start/End Out)
- 4 MDOs (Message Data Out) or Aurora Output
- One MCKO (Message Clock Out)

The output ports are synchronized to the Nexus output clock (MCKO).

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see Section 11.9.3.2, IEEE 1149.1 (JTAG) Test Access Port for details). A separate JTAG TAP controller is instantiated within the NXMC.

11.11.2.3 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in the following table. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 11-25. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 36
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 11-26)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 37
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 11-26)
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (See Table 11-27)
	14	14	ECODE	fixed	Error code (See Table 11-28)
Data Trace - Data Write	6	6	TCODE	fixed	TCODE number = 38

Table continues on the next page...

Table 11-25. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Message w/ Sync	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 11-26)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 39
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 11-26)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	8	8	WPHIT	fixed	Watchpoint source(s) number (See Table 11-35)

Table 11-26. Data trace size (DSZ) encodings (TCODE = 36,37,38,39)

DSZ	Transfer size
001	8-bit

Table continues on the next page...

Table 11-26. Data trace size (DSZ) encodings (TCODE = 36,37,38,39) (continued)

DSZ	Transfer size
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 11-27. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110-1111	Reserved

Table 11-28. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description
13 to 8	7 to 0	
Source-ID (SRC): NXMC0 = 0xC, NXMC1 = 0xD		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved

11.11.2.4 NXMC register map

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See IEEE 1149.1 (JTAG) Test Access Port for details. The complete memory map appears in the following table.

Table 11-29. NXMC register map

Nexus register	Nexus access opcode	Read/ write	Read address	Write address
Development Control 1 (DC1)	$2	R/W	$04	$05
Development Control 2 (DC2)	$3	R/W	$06	$07
Watchpoint Trigger (WT)	$B	R/W	$16	$17
Data Trace Control (DTC)	$D	R/W	$1A	$1B
Data Trace Start Address1 (DTSA1)	$E	R/W	$1C	$1D
Data Trace Start Address2 (DTSA2)	$F	R/W	$1E	$1F
Data Trace End Address1 (DTEA1)	$12	R/W	$24	$25
Data Trace End Address2 (DTEA2)	$13	R/W	$26	$27
Breakpoint/Watchpoint Control Register1 (BWC1)	$16	R/W	$2C	$2D
Breakpoint/Watchpoint Control Register2 (BWC2)	$17	R/W	$2E	$2F
Breakpoint/Watchpoint Address Register1 (BWA1)	$1E	R/W	$3C	$3D
Breakpoint/Watchpoint Address Register2 (BWA2)	$1F	R/W	$3E	$3F
Reserved	$20-$3F	-	$40-$7E	$41- $7F

11.11.2.5 NXMC register descriptions

Detailed register definitions for the current NXMC implementation are described in the chapter Nexus Crossbar Multi-Master Client (NXMC).

11.11.3 Functional description

11.11.3.1 Programming considerations (RESET)

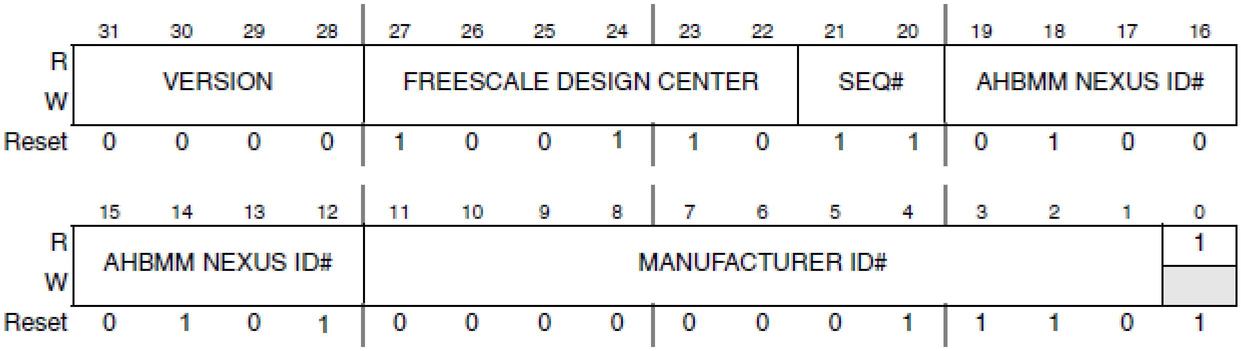
If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of TRST and system reset de-assertion, after the JTAG ID register has been read by the tool.

11.11.3.2 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in NXMC register access via JTAG.

11.11.3.3 JTAG ID register

This JTAG ID Register (included in the NXMC module) provides key development attributes to the development tool concerning the NXMC block. This register is fixed and is accessed through the standard JTAG IR/DR paths.



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	VERSION				FREESCALE DESIGN CENTER						SEQ#		AHBMM NEXUS ID#			
Reset	0	0	0	0	1	0	0	1	1	0	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AHBMM NEXUS ID#				MANUFACTURER ID#											1
W																
Reset	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1

Figure 11-11. JTAG ID register

The following table describes the JTAG ID register fields.

Table 11-30. JTAG ID field descriptions

Field	Description
31:28	Embedded Product Version Number, set to '0000'
27:22	Freescale Design Center ID Number, set to restore '100110'
21:20	Embedded Product Sequence Number, set to '11'
19:12	AHBMM Nexus Module ID, set to '01000101'
11:1	Manufacturer ID Number, set to '0000001110'
0	Fixed per JTAG 1149.1

11.11.3.4 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction (NEXUS-ACCESS =0x0) into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The NXMC module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the TRST pin or by cycling through the state machine using the TMS pin. The NXMC module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided, and the module disables the (drive inactive) auxiliary port output pins (MDO[30:0], MSEO[1:0], MCKO). NXMC registers are not available for reads or writes.

11.11.3.5 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the TRST pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set, etc.

11.11.3.6 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction (NEXUS-ACCESS = 0x0) into the JTAG Instruction Register (IR).

Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to Table 11-29.

Reading/writing of a NXMC register then requires two passes through the Data-Scan (DR) path of the JTAG state machine. See details about the IEEE 1149.1-2001 TAP Controller Finite State Machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table 11-29) and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown in the following figure and table.

	7	6	5	4	3	2	1	0
R W	Nexus Register Index							R/W
Reset	0	0	0	0	0	0	0	0

Table 11-32. JTAG DR field description for Nexus register access

Field	Description

Table continues on the next page...

Table 11-32. JTAG DR field description for Nexus register access (continued)

Nexus Register Index	Selected from values in Table 11-29
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
    a. During a read access, data is latched from the selected Nexus register when the JTAG state machine (see IEEE 1149.1-2001 TAP Controller Finite State Machine) passes through the Capture-DR state.
    b. During a write access, data is latched into the selected Nexus register when the JTAG state machine (see IEEE 1149.1-2001 TAP Controller Finite State Machine) passes through the Update-DR state.

11.11.3.7 Data trace

This section describes with the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001 standard.

11.11.3.7.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

11.11.3.7.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

11.11.3.7.3 Enabling DTM operation

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register via JTAG to enable data trace.
- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC1 register to enable data trace. (The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

NOTE

In case of enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (i.e. DTC, DTSA/E1/2 and WT register fields) needs to be configured with right values in order to ensure correct DTM functioning.

11.11.3.7.4 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

NOTE

If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > DTM).

11.11.3.7.5 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

11.11.3.7.6 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

11.11.3.7.7 System bus cycle special cases

The system bus cycle special cases are shown in following table.

Table 11-33. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

11.11.3.7.8 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

11.11.3.8 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2003. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

11.11.3.8.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC1 Register. This bit is set either via JTAG write 1 access and cleared via JTAG write 0 access.

NOTE

> In case of enabling the watchpoint messaging, it is important to make sure that all the other Watch Point Message (WPM) controls (i.e. BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported by using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a AHB access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too, as explained for DTM (in "System bus multi-master tracing").

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When a watchpoint source asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in the following figure and table.

Table 11-34. Watchpoint message format (fixed length = 18 bits)

8 bits	4 bits	6 bits
WPHIT (RRRRXXXX)	SRC	TCODE (001111)

Table 11-35. Watchpoint source description

Watchpoint source (8 bits)	Watchpoint description
RRRRXXX1	reserved
RRRRXX1X	reserved
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

11.11.3.8.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in Figure 29.

Table 11-36. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

Chapter 12
Power Management

12.1 Overview

MPC5744P microcontroller includes a robust power management infrastructure that enables applications to select among various operational and low-power modes and to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can leave RESET. This chapter gives an overview of the built-in power management features.

The power management infrastructure comprises the following modules:

- Power Control Unit (MC_PCU)
- Power Management Controller (PMC)

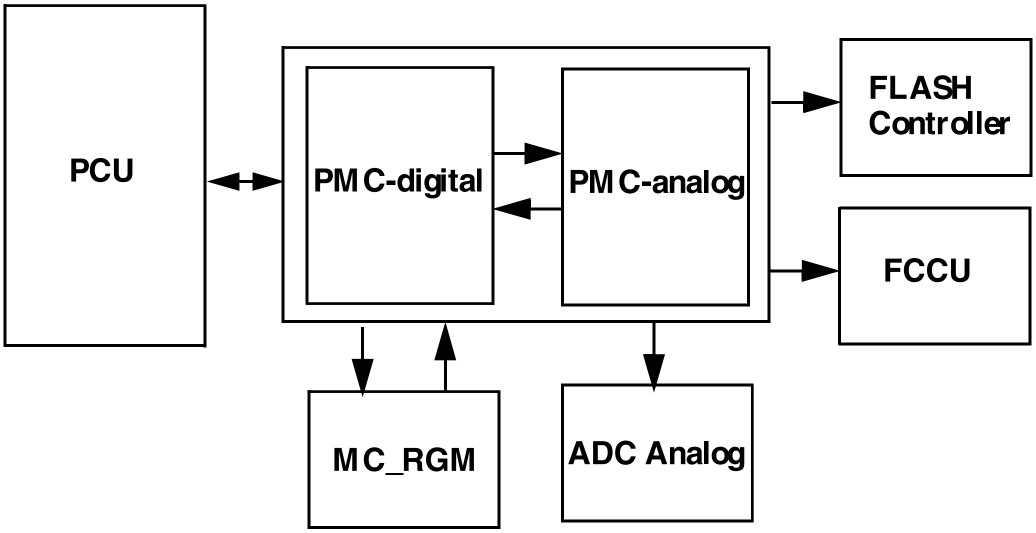


Figure 12-1. Power Management Overview

This chapter describes the modules interaction in brief. For complete details, see the respective module chapters.

12.1.1 Power Control Unit (PCU)

The Power Control Unit (PCU) is used mainly to map the Power Management Controller (PMC) registers into the PCU address space. These registers control the different low voltage detect options and power options for the device. On the MPC5744P device there is only one single power domain, therefore reducing power by switching off different areas of the device is not an option. There are low power modes which can be used, namely HALT and STOP for this purpose.

12.1.2 Power Management Controller (PMC)

The Power Management Controller implements a linear voltage regulator to generate the core logic supply from a 3.3 V (nominal) input supply and monitors various supply voltages of the MPC5744P chip for functional safety purpose. Depending on the supply voltages, power-on-reset (POR) and destructive resets are generated.

The PMC connects to the Reset Generation Module (MC_RGM), the Fault Collection Control Unit (FCCU) and the flash memory controller. The on-chip ADC receives signals from PMC for diagnostic purposes.

12.1.3 Supply Concept

MPC5744P provides on-chip regulator to derive the core supply from external 3.3 V supply. A power transistor must be added on the PCB. See the MPC5744P Data Sheet for specifics on this device. Alternatively, an external linear or switching regulator can be used. The analog supply needs to be protected from noise on digital supply at least by means of decoupling capacitor. Better filters can be used if they do not affect regulator stability. A separate regulator for analog supply is preferred.

The PMC monitors each supply voltage and releases the resets when they are within the limits of operating conditions. The power-on-reset can be generated on-chip or can be provided from external source, such as brown-out device.

Some supplies may already be connected internally inside the package depending on the package type, mainly in order to save pins in low pin-count packages.

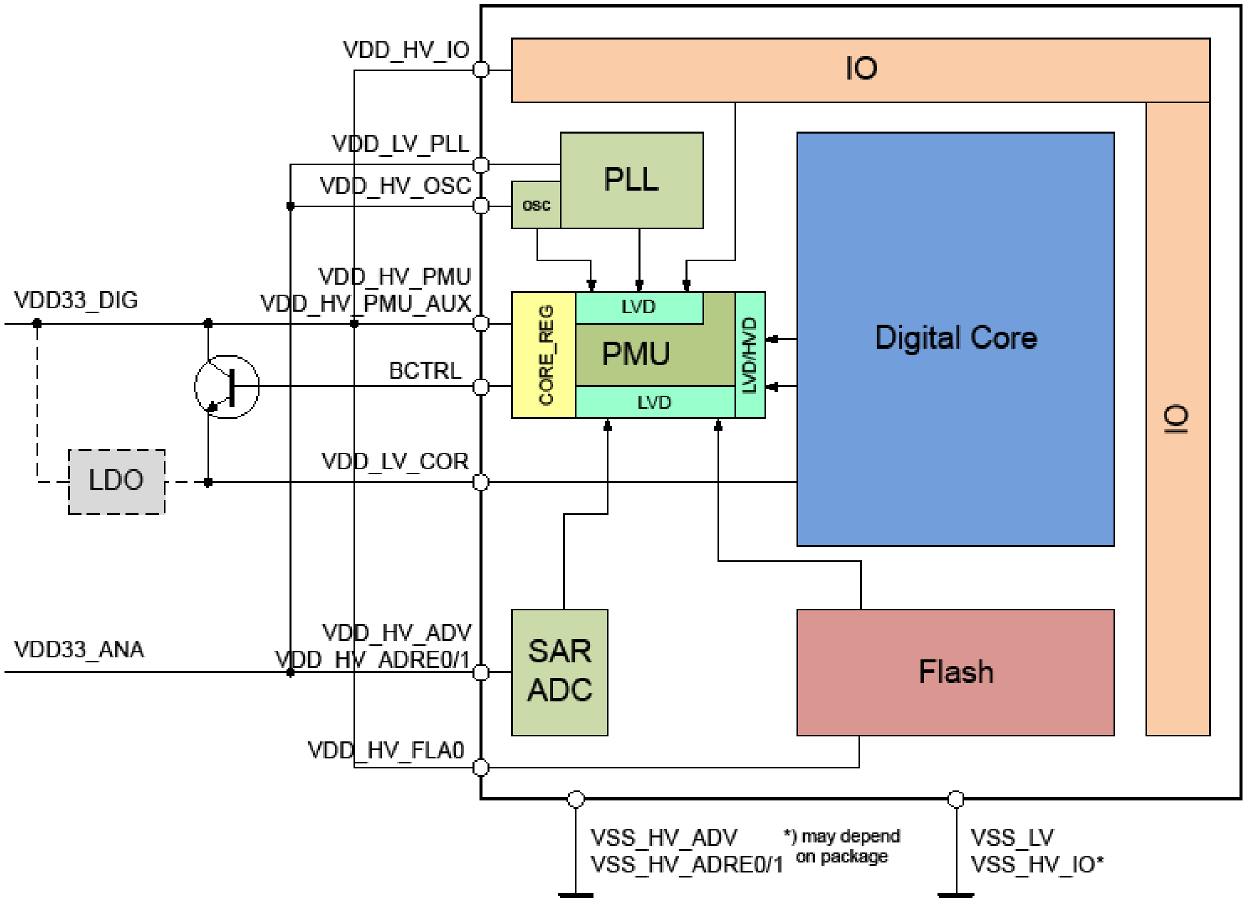


Figure 12-2. Supply concept

12.1.4 Power Modes

MPC5744P does not support ultra-low power modes that require power gating or separate power domains. The core logic implements stop and low frequency power saving modes, which may result in core current steps. The effect needs to be mitigated by a smoother frequency ramps instead of instantaneous change to avoid triggering LVD or HVD by under/overshoots.

12.1.5 External Supply

The internal supply regulator becomes inactive when an external Low Dropout Register (LDO) is used. The Bipolar Junction Transistor (BJT) should be removed in that case. It can be assumed that the internal regulator has no undesirable impact if ballast BJT is not present. The internal regulator can be switched off by setting the PMC module's PMCCR[INT_REG_BYPASS] field to 1.

12.1.6 Power Management Supply Description

The following table lists the PMC supply signals.

Table 12-1. Power management controller external signals

Name	Type	Description
VDD_HV_ADRE0	Reference	ADC0 High reference voltage
VDD_HV_ADRE1	Reference	ADC1 High reference voltage
VDD_HV_ADV	Supply	High voltage supply for the ADC modules
VDD_HV_IO	Supply	High voltage Power supply for the I/Os
VDD_HV_PMC	Supply	High voltage power supply for internal power management unit
VDD_HV_OSC0	Supply	High voltage power supply for the internal oscillator
VDD_LV_LFAST	Supply	Low voltage power supply for the LFAST module
VDD_LV_PLL	Supply	Low voltage power supply for the PLL module
VDD_LV_NEXUS	Supply	Low voltage power supply for the Nexus module
VSS_LV	Ground	Ground supply for the device / I/O. This is covering both VSS_LV and VSS_HV in case of exposed pad device.
VSS_LV_LFAST	Ground	Ground supply for the LFAST module
VSS_LV_NEXUS	Ground	Ground supply for the Nexus module
VSS_HV_IO	Ground	Ground supply for the device I/O.
VSS_HV_OSC0	Ground	Ground supply for the internal oscillator module
VSS_HV_ADRE0	Ground/ Reference	ADC0 Ground and low reference voltage
VSS_HV_ADRE1	Ground/ Reference	ADC1 Ground and low reference voltage
VSS_HV_ADV	Supply	Ground supply for the ADC modules

12.1.7 Voltage Monitoring

The PMC senses the supply voltages continuously to keep the chip in safe operating range. Destructive resets or alternatively, interrupts are triggered, when the voltage exceeds or falls below the limits defined by recommended electrical operating conditions.

12.1.7.1 Core Supply Monitor

The digital core supply range is compared with a low and high voltage limit.

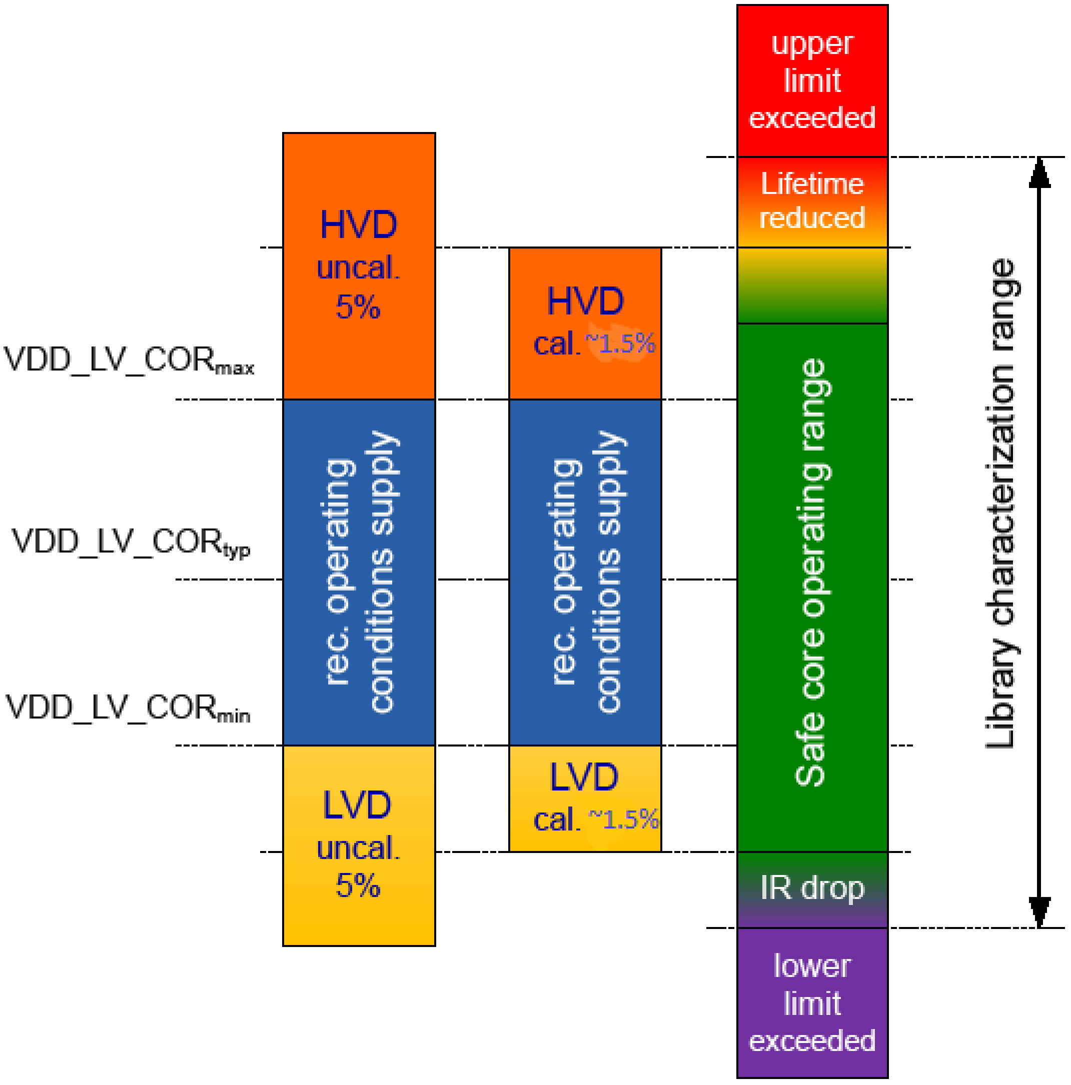


Figure 12-3. Core Supply Monitoring

The digital core is able to operate between the limits given by the MPC5744P Data Sheet. The lower limit is reduced by the amount of on-chip IR drop, that must be granted. The upper limit is defined by the beginning of lifetime degradation.

The available range for external supply voltage is further restricted by the tolerance of high (HVD) and low voltage monitors (LVD). See the MPC5744P Data Sheet for exact values. To ensure safe operating conditions for core the LVDs/HVDs are calibrated. Calibration of LVD/HVD takes places after the core is in functional safe operating range. It is performed in a way that LVD/HVD cover the allowed core operating range, i.e. by increasing the minimum value for LVD and decreasing the maximum value for HVD.

HVD checks the core supply for exceeding an upper limit. It has a 5% uncalibrated tolerance that can be reduced to ~1.5% by calibration. LVD checks the core supply for exceeding a lower limit. It has a 5% uncalibrated tolerance that can be reduced to ~1.5% by calibration.

If LVD and HVD are disabled, the external core supply can have an extended operating range. External voltage monitors shall be used in that case to ensure functional safety.

The core LVD/HVDs are duplicated for use at two different design points (LVD_CORE and LVD_CORE_BK or HVD_CORE and HVD_CORE_BK). The core supply is sensed at one external supply feed point and the IR drop critical point.

The core LVD/HVDs shall provide sufficient hysteresis or low-pass characteristic to prevent bouncing in case of short core supply over/undershoots.

12.1.7.2 IO, REG and other 3.3 V Supplies Monitor

All other 3.3 V, non-core supplies are compared to a lower limit only by means of LVDs. The following table lists the LVDs for the MPC5744P device.

All 3.3 V supplied modules must remain fully functional in an extended operating range that includes the LVD calibrated tolerance. The 3.3 V LVDs have 5% uncalibrated and 2% calibrated tolerance. Calibration takes place when the core supply is within a safe operating range. After calibration, the typical LVD value is centered within the recommended operating conditions low limit and the safe operating conditions low limit to provide margins for supply undershoots.

If LVDs are disabled, external voltage monitors shall be used to ensure functional safety. The LVDs shall provide sufficient hysteresis or low-pass characteristic to prevent bouncing in case of short 3.3 V supply undershoots.

Table 12-2. List of 3.3 V Low Voltage Detect Monitors

LVD Supply Monitored	Voltage Supply
LVD IO	3.3 V
LVD VDDREG	3.3 V
LVD VDDFLASH	3.3 V
LVD VDDADC	3.3 V

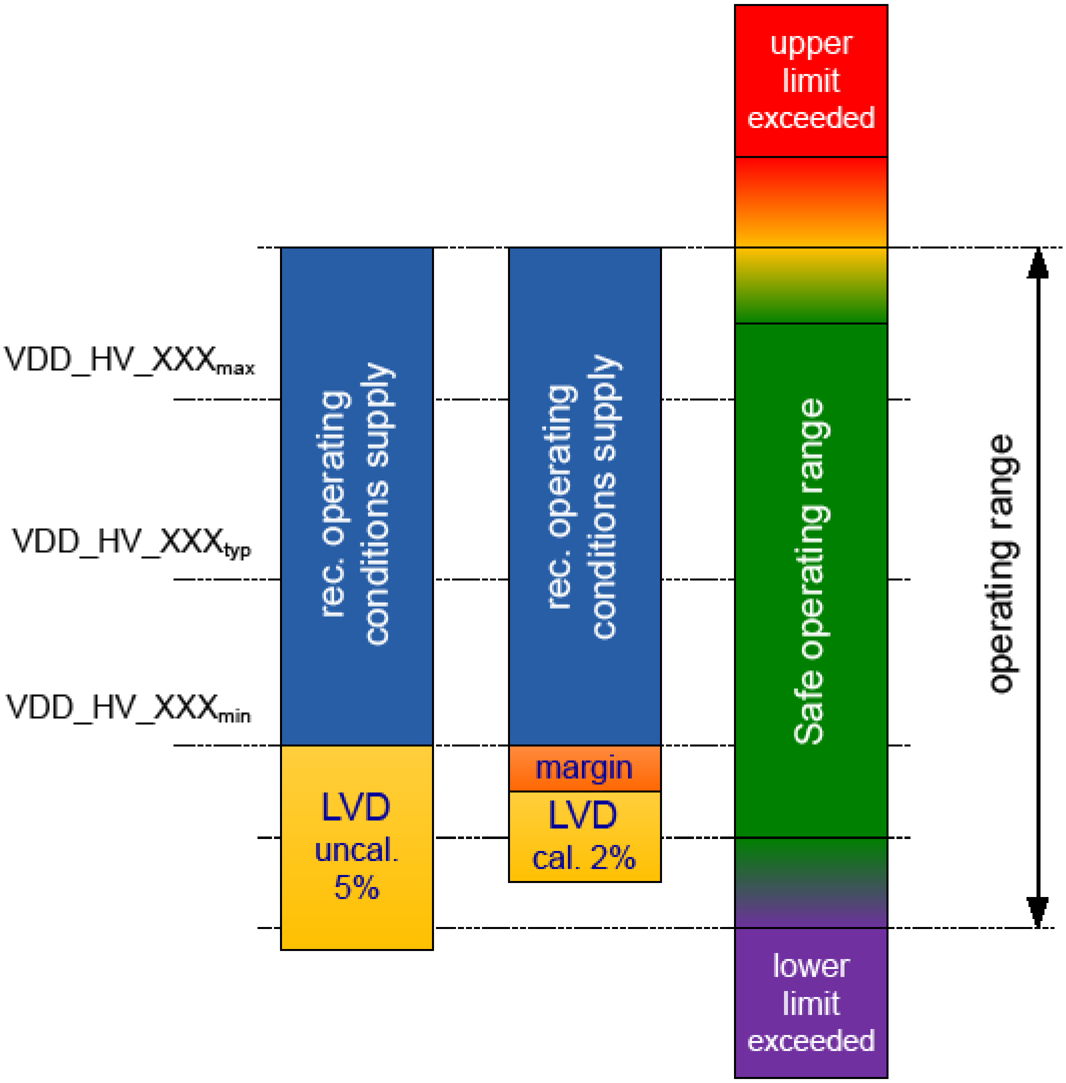


Figure 12-4. 3.3V Supply Monitoring

12.1.8 Power-on Reset

The Power-on Reset (POR) Generator is required to initialize the chip during supply rise. It is a voltage monitor which detects if a supply exceeds a defined threshold voltage level. The POR checks the core voltage supply VDD12_COR and the regulator supply VDD_HV_REG. When both supplies have a reliable level for the operation of the PMC and RESET generator, the LVDs/HVDs are activated.

The POR watches the rising edge of the supply before the recommended operating conditions are reached. The POR functions correctly even if the supply voltage is non-monotonic. It includes sufficient hysteresis to avoid bouncing in case of slow supply ramps. Within the safe operating conditions, the LVDs/HVDs take control over reset generation.

The POR must hold the device in reset regardless of how slow the supply rise time is, until the point at which the POR is designed to release. The rate of rise of a battery supply is described in an industry standard specification, ISO 16750-2. The slow ramp up / ramp down of battery supply and following VDD is specified as 0.9 V / sec.

12.1.9 Power-up Sequence

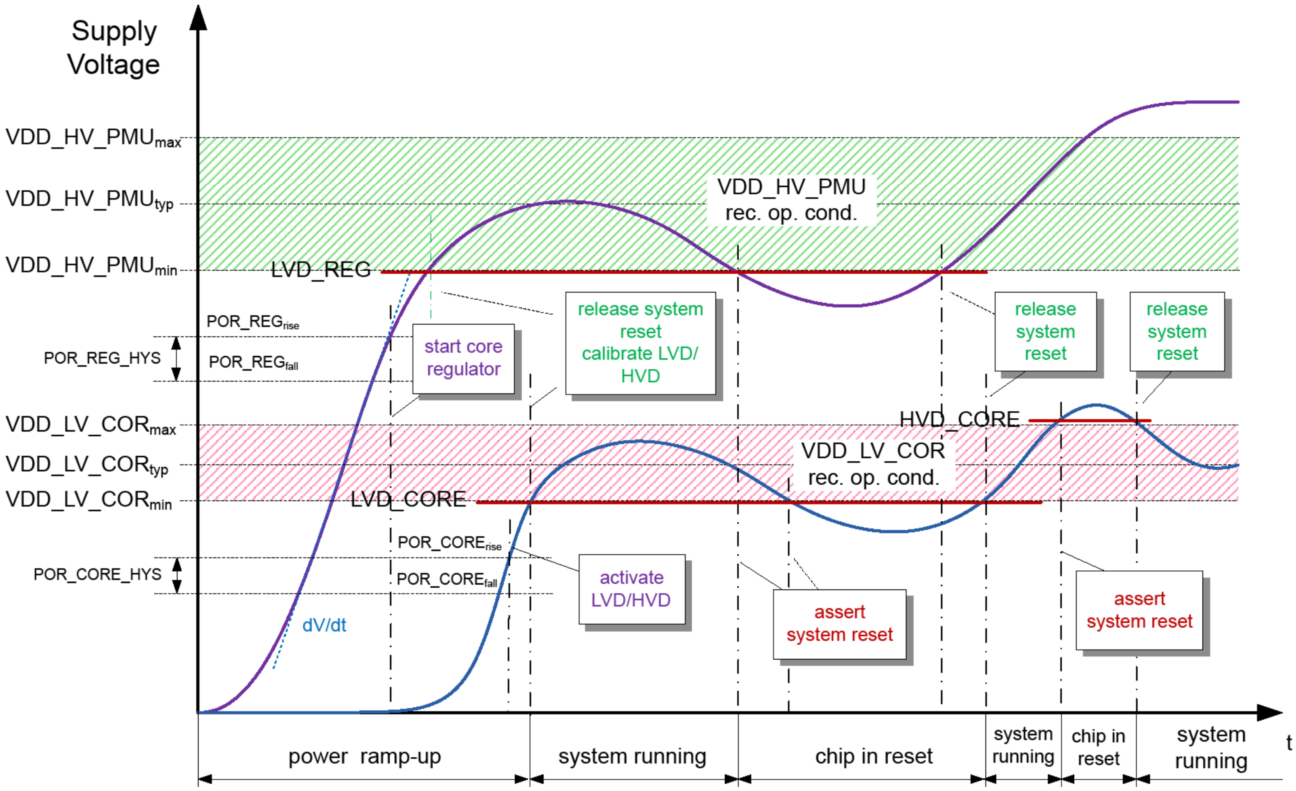


Figure 12-5. OFLAG timing during non-continuous counting

The power-up sequence runs whenever the system will get through a power cycle or external reset.

When the VDD33_REG voltage exceeds the POR_REG rising edge threshold, all 3.3V LVDs are reset to their uncalibrated state and armed. The core voltage regulator CORE_REG starts to work and the core supply voltage rises.

When the core voltage crosses the POR_CORE rising edge threshold, the core LVDs and HVDs are reset to their uncalibrated state and armed.

When the core supply and the regulator voltage reach their minimum value according to the recommended operating conditions, the system reset is released and the system boot process starts. The LVDs/HVDs are now calibrated to provide tighter limits for voltage over/undershoot detection.

As soon as core or any 3.3V supply falls below the limit of the corresponding LVD thresholds, a destructive reset is triggered. The system is held in reset state until all LVD threshold levels are exceeded again. If the core voltage exceeds the threshold of the HVD, a destructive reset is asserted. It is released when the core voltage falls below the HVD threshold.

When core or regulator supply fall below the POR_CORE or POR_REG falling edge limit, respectively, the system is reset and core supply is disabled. The chip will restart with a complete power-up cycle.

12.1.10 ADC Interface

The ADC interface provides the PMC internal voltages for diagnostic purpose. The analog multiplexer and the buffer is inside the PMC. The SAR ADC converts the selected voltage, which can then be evaluated by the CPU.

12.1.11 Safety Measures

The PMC implements a self-test architecture. The self-test can be triggered by test program or software. It is executed automatically with each power-cycle. The self-test will generate non-critical faults, which are collected in the FCCU module. The LVD/HVDs allow also error injection by SW, for artificially triggering an error in FCCU.

Parts of the PMC design that can act as a single point of failure are replicated, such as the analog voltage reference generation circuits.

Various internal voltages can be monitored during PMC operation, with assistance of the ADC.

The PMC provides a signal that keeps the I/O pads in a reset condition if either the core supply or the IO supply is out of the valid operating range.

Chapter 13
Clocking

13.1 Introduction

This chapter describes the architecture for the system level clocks and includes the following information:

- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers
- Progressive Clock Switching

NOTE

This chapter covers only clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.

The clock (system clock) for the cores, memories, and debug logic is independent from the peripheral clocks. This independence allows the system clock to be a frequency-modulated clock to reduce electromagnetic emissions while preserving a precise clock for peripheral timer and communication functions. It also allows the system clock to be reduced during low computational activity periods while maintaining timer and communication link operation.

The MPC5744P boots from the 16 MHz internal RC oscillator (IRCOSC) and uses it as a backup clock (if a backup clock is enabled) in the event of a PLL or oscillator failure.

There are three possible ways to provide the source clock:

- External oscillator
- External crystal
- 16 MHz internal RC oscillator

From one of these input sources, the internal clocks are generated from one of the two PLLs, PLL0 and PLL1, PLL0_PHI and PLL1_PHI, respectively. These two clocks, along with the XOSC and IRCOSC, can be selected to drive system peripherals depending on the configuration of the Auxiliary Clock Selectors. The PHI1 output of PLL0 can also be used as the clock source for PLL1. A total of seven clock selectors allows developers to select the PLL reference clocks, drive various system peripherals with an independent clock source, and select a clock source to drive the CLKOUT signal for off-chip use. There is also one additional clock selector which is used exclusively for the system clock.

Each of the outputs of the module clock selectors has up to three dividers, which allows for even more clock frequency granularity with division factors up to 64 for a given group of peripherals. On selected outputs of the clock dividers are Clock Monitor Units (CMU) that are used to test the integrity of the clocks making sure that their frequencies stay within necessary operating limits. If any of the five CMUs detects an issue with the clock signal that is being monitored, an interrupt or system reset could be generated, depending on how the CMUs are configured.

13.2 Clock generation

The top-level clock generation architecture appears in the following figure. All clock selectors and dividers in the figure are located and programmed in the Clock Generation Module (MC_CGM). All dividers are integer dividers (1, 2, 3, …, n) with the ranges given in the diagram.

NOTE

CMU0…CMU4 in the figure use IRCOSC for their reference clock. However, the user can select XOSC as the reference in CMU0 for frequency measurement of IRCOSC.

Note: All dividers shown in the diagram (FCD not included) are integer dividers with a range of 1, 2, 3,...., n.
All clock dividers are 50% duty cycle.

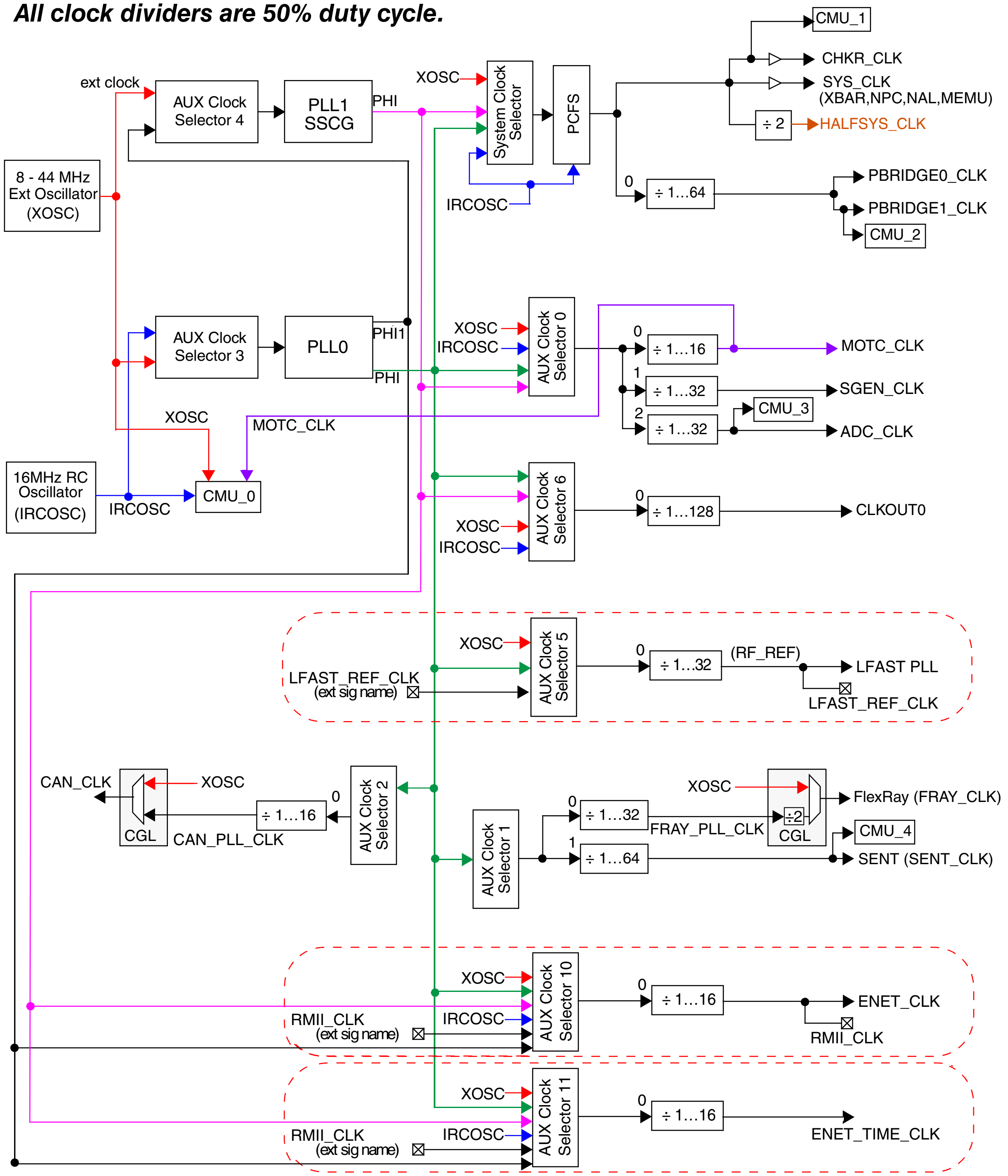


Figure 13-1. Clock generation

NOTE

PLL1 is routed on N15P devices into the AUX Clock Selector 0. It is recommended not to enable Frequency modulation on

PLL1 when it is being used as a clock source for AUX Clock Selector 0.

NOTE

See Figure 13-4 for LFAST clocking details.

13.2.1 MC_CGM registers

The following table shows the relationship between the clocks in Figure 13-1 and the MC_CGM registers.

NOTE

See the Clock Generation Module (MC_CGM) for specifics on register implementation.

Table 13-1. MC_CGM relationship to clocks

MC_CGM registers	Register description	Output clock	CMU
—	—	Checker Core Main Core RCCU_0 SYSCLK (XBAR, NPC, NAL, MEMU)	CMU_1[1]
		HALFSYS_CLK (SYSCLK ÷ 2)	-
MC_CGM_SC_DC0	System Clock Divider Configuration 0	PBRIDGE0_CLK PBRIDGE1_CLK	CMU_2[2]
MC_CGM_SC_DC2	Not used	—	—
MC_CGM_AC0_DC0 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	MOTC_CLK	CMU_0
MC_CGM_AC0_DC1 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 1 Aux Clock 0 Select Control	SGEN_CLK	—
MC_CGM_AC0_DC2 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	ADC_CLK	CMU_3
MC_CGM_AC1_DC0	Aux Clock 1 Divider Configuration 0	FRAY_PLL_CLK	—
MC_CGM_AC1_DC1	Aux Clock 1 Divider Configuration 1	SENT (SENT_CLK)	CMU_4
MC_CGM_AC2_DC0	Aux Clock 2 Divider Configuration 0	CAN_PLL_CLK	—
MC_CGM_AC3_SC	Aux Clock 3 Select Control	PLL0	—
MC_CGM_AC4_SC	Aux Clock 4 Select Control	PLL1	—
MC_CGM_AC5_DC0 MC_CGM_AC5_SC	Aux Clock 5 Divider Configuration 0 Aux Clock 5 Select Control	RF_REF (LFAST)	—
MC_CGM_AC6_DC0	Aux Clock 6 Divider Configuration 0	CLKOUT0	—

Table continues on the next page...

Table 13-1. MC_CGM relationship to clocks (continued)

MC_CGM registers	Register description	Output clock	CMU
MC_CGM_AC6_SC	Aux Clock 6 Select Control		
MC_CGM_AC10_DC0 MC_CGM_AC10_SC	Aux Clock 10 Divider Configuration 0 Aux Clock 10 Select Control	ENET_CLK	—
MC_CGM_AC11_DC0 MC_CGM_AC11_SC	Aux Clock 11 Divider Configuration 0 Aux Clock 11 Select Control	ENET_TIME_CLK	—

1. Connected to the RCCU clock path (which is part of the Checker Core clock), after the branch of the clock line for Master and Checker Core and after the branch between Checker Core clock and RCCU clock.
2. Connected to the PBRIDGE1_CLK (downstream from the common part of the clock tree for the Peripheral Bridges)

13.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in the following table.

NOTE

The user is responsible to verify that these values are not exceeded.

NOTE

Each AIPS Bridge (PBRIDGE_0, PBRIDGE_1) is capable of clocking the AIPS interface of each peripheral slot individually at the XBAR frequency, or at the XBAR frequency divided by two.

Table 13-2. Maximum system level clock frequencies

System clock	Max frequency (MHz)
Cores, NPC, NAL, MEMU (CHKR_CLK, SYS_CLK)	200
XBAR (SYS_CLK)	200
PBRIDGE_0, PBRIDGE_1, SIPI, DMA_CH_MUX	50
Motor control (MOTC_CLK)	160
DMA, Interrupt Controller (HALFSYS_CLK)	100
ADCs	80
Sine Wave Generator (SGEN)	20
LFAST	320
FlexRay (FRAY_CLK)	80
FlexCAN (CAN_CLK)	80
SENT (SENT_CLK)	80
Ethernet (AHB clock)	100

Table continues on the next page...

Table 13-2. Maximum system level clock frequencies (continued)

System clock	Max frequency (MHz)
ENET_CLK	50
ENET_TIME_CLK	50 (integer ns period requirement)

In order to maintain synchronization between the different system clock branches (for example, SYS_CLK output from the System Clock Selector), there are limitations on the frequencies of those clocks. The system clocks must be binary multiples of each other with the frequency relationships described in the electrical specifications.

13.3.1 JTAG Frequencies

The following table shows the maximum frequencies of the JTAG interface.

Table 13-3. JTAG Frequencies

Configuration	e200z4 Max Internal JTAG Frequency	Maximum External JTAG Frequency	Notes
Unconfigured clock	-	See electrical characteristics for maximum usable external JTAG frequency	Assumes 16 MHz clock with +/- 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
Clock configured to 180 MHz	90 MHz		JTAG usable at 1/2 CPU frequency

13.4 Default clock configuration

At power up, the IRCOSC is the default clock for the entire system. All system clock dividers are set to divide by two at power-on reset.

13.5 Clock sources

The following clock sources are described in this section:

- PLL0
- PLL1 (FMPLL)
- External Oscillator (XOSC)
- External Clock (EXTAL Bypass)
- 16 MHz Internal RC Oscillator (IRCOSC)

13.5.1 PLL

The PLL system in the MPC5744P is a Dual PLL that provides separate system and peripheral clocks. The PLLs are disabled after power on and must be enabled by software. The block diagram for the Dual PLL appears in the following figure.

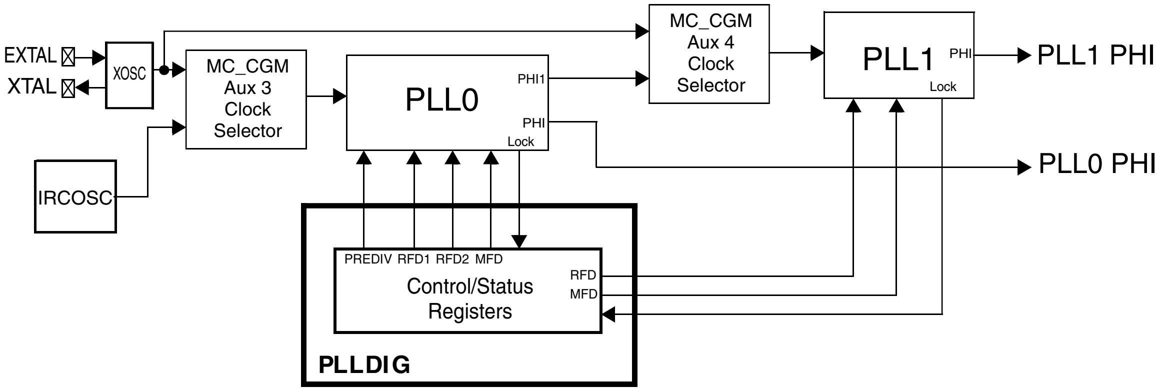


Figure 13-2. MPC5744P Dual PLL digital interface block diagram

13.5.1.1 PLL0 - base PLL (non-FM)

PLL0 is the primary PLL. This PLL is used to source a non-Frequency Modulated clock to the MPC5744P modules and also the reference clock to PLL1.

13.5.1.1.1 PLL0 input clocks

The possible input clock sources for PLL0 are the XOSC, IRCOSC, and EXTAL Bypass. The EXTAL Bypass input is the EXTAL pin. AUX Clock Selector 3 selects which input clock will be used as the source for PLL0. The selection between XOSC and EXTAL Bypass is made via the XOSC_CTL register of the XOSC module.

13.5.1.1.2 PLL0 output clocks

The output clocks from PLL0 are PHI and PHI1. The PHI output clock drives various peripheral clocks and the system clock when selected in the MC_CGM. The PHI1 output provides one of the input references for PLL1.

13.5.1.2 PLL1 - FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is typically used to drive the system clock. PHI is the output of PLL1 which drives the System Clock Selector and AUX Clock Selector 6 of the MC_CGM.

13.5.1.2.1 PLL1 input clocks

The possible input clock sources for PLL1 are XOSC, PLL0_PHI, and EXTAL Bypass. The EXTAL Bypass input is the EXTAL pin, which "bypasses" the XOSC output. AUX Clock Selector 4 selects which input clock is used as the source for PLL1. The selection between XOSC and EXTAL Bypass is made via the XOSC_CTL register of the XOSC module.

13.5.1.2.2 PLL1 output clocks

The output clock from PLL1 is the PHI clock, which can drive the system clock if the System Clock Selector of the MC_CGM is configured to do so. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

13.5.1.2.3 PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of the PLLs. Details are provided in the "Dual PLL Digital Interface" chapter.

13.5.1.3 PLL register reset values

All reset values for the PLLDIG registers that are device specific are shown in the following table. All other register reset values are shown in the Dual PLL Digital Interface chapter.

Table 13-4. Dual PLL Digital Interface register reset values

Offset (hex)	Register	Reset Value (hex)
0008	PLL0DV - PLL0 Divider Register	0000_0000h
0028	PLL1DV - PLL1 Divider Register	0000_0000h

13.5.1.4 PLLDIG initialization information

Coming out of reset PLL0 and PLL1 are disabled per the DRUN mode configuration register, MC_ME_DRUN_MC.

1. Configure PLL0 and related modules.
    a. With PLL0 disabled, program PLL0 clock source in MC_CGM_AC3_SC[SELCTL]. Default source is 16 MHz IRCOSC. Write MC_CGM_AC3_SC[SELCTL] = 1 to select XOSC as source.

    b. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].

    c. If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the MC_ME).

    d. If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.

2. Turn on XOSC and PLL0.
    a. Configure a mode configuration for turning on PLL0 and XOSC. In a mode configuration register (for example, MC_ME_DRUN_MC) set MC_ME_<*mode*>_MC[XOSCON] = 1 and MC_ME_<*mode*>_MC[PLL0ON] = 1. If desired, also set MC_ME_<*mode*>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0 (PLL0:PHI output) as the sysclk.

    b. Enter that mode by two writes to MC_ME_CTRL register to enter that mode. This is required even if entering the same mode.

3. Wait for the mode transition to complete.
    a. Wait for the mode transition to complete by polling MC_ME_GS[S_MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

        - XOSC counter expires (if the new mode configuration changes MC_ME_<*mode*>_MC[XOSCON] = 1), and
        - PLL is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLL0ON] = 1)

   b. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

4. Configure PLL1 and related modules.

   a. With PLL1 disabled, program PLL1 clock source in MC_CGM_AC4_SC[SELCTL]. Default is MC_CGM_AC4_SC[SELCTL] = 1 which selects XOSC.

   b. Program PLL1DV[MFD] and PLL1DV[RFDPHI].

   c. If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.

5. Turn on PLL1.

   a. Configure a mode configuration for turning on PLL1, and keeping XOSC and PLL0 on. In a mode configuration register (for example, MC_ME_DRUN_MC), initialize MC_ME_<*mode*>_MC[XOSCON] = 1, MC_ME_<*mode*>_MC[PLL0ON] = 1 and MC_ME_<*mode*>_MC[PLL1ON] = 1. If desired, also set MC_ME_<*mode*>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.

   b. Enter that mode by two writes to ME_CTRL register to enter that mode. This is required even if re-entering the same mode.

6. Wait for the mode transition to complete.

   a. Wait for the mode transition to complete by polling MC_ME_GS[S_TRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

      - XOSC counter expires (if the new mode configuration changes XOSCON to 1), and

      - PLL1 is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLL1ON] = 1).

   b. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

NOTE

See the "Mode Entry Module (MC_ME)" chapter in this *Reference Manual* more details.

13.5.2 External oscillator (XOSC)

The external oscillator (XOSC) is provided as a reference for the on-chip PLLs. It can also can be used as a clock source to the ADCs, SGEN, motor control modules, LFAST, and PLL reference as well as a system clock source. It is available for observation on either of the CLKOUT pins, and is used as a reference to calibrate the IRCOSC frequency.

The external oscillator allows a crystal or external clock to be used as the reference clock for the microcontroller. The XOSC has the following features:

- Reference clock to PLL0 and PLL1
- Reference clock to CMU0 (IRCOSC trimming)
- Option to drive the CAN and FlexRay protocol clocks directly from the XOSC
- Provides support for 8 MHz to 44 MHz crystal inputs

13.5.2.1 XOSC reset value

The following table shows the default reset value for the XOSC registers.

Table 13-5. XOSC register reset values

Offset	Register	Reset Value
0x00	XOSC_CTL-XOSC Control Register	0030_0000h

13.5.3 16 MHz internal RC oscillator (IRCOSC)

The microcontroller has a 16 MHz internal RC oscillator which is always enabled and can be used as the clock source for the PLLs. It is used as the clock source for the MC_RGM, FCCU, PIT, SWT, and SIUL2 input filters, and it can also be used as a clock source for the ADCs, SGEN, motor control modules, and system clock. The IRCOSC is the default system clock after reset. The register interface is used for user trimming of the oscillator.

13.5.3.1 IRCOSC register interface

This device has a dedicated digital interface for the IRCOSC. The digital interface provides a register for fine tuning the IRCOSC frequency by the user. Other registers allow reading of the settings of IRCOSC temperature sensor, voltage regulator and capacitor trimming values that are set by the factory. The IRCOSC is always enabled when the device is powered.

13.5.3.2 IRCOSC trimming

The IRCOSC requires trimming to meet accuracy requirements. During the initial phases of a reset sequence triggered by a power-on or destructive reset, the IRCOSC runs at its untrimmed frequency value. A trim value that was determined during factory test, and subsequently stored in flash memory, is loaded during reset via the SSCM to the IRCOSC_CTL[USER_TRIM] (see the "IRCOSC Digital Interface" chapter for details).

13.6 Peripheral clocks

The following figure shows the clock distribution to the core and peripheral modules.

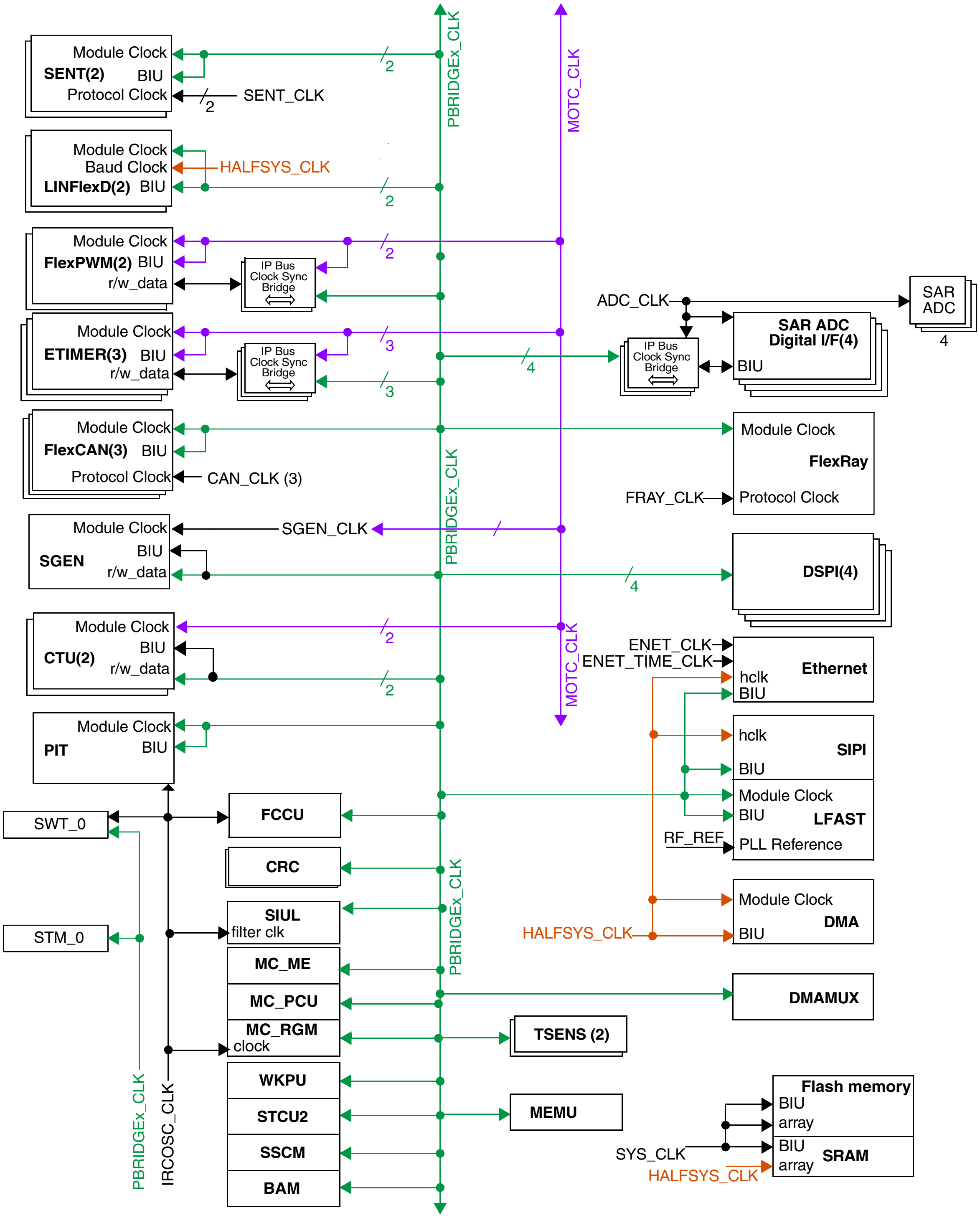


Figure 13-3. Clock distribution

NOTE

Before Disabling the clock from root (source), make sure that all the peripherals that are using the clock are switched OFF or at least that clock is not selected as source of Peripheral clock. For Example:- In FlexCAN when XOSC is selected for the

CAN Engine Clock Source (FLEXCAN_CTRL[CLKSRC]), before disabling FlexCAN Peripheral Clock, the user must ensure XOSC is enabled during the target mode transition i.e. XOSC must be enabled for the target mode. After the FlexCAN Peripheral Clock is switched OFF user can disable XOSC clock.

13.6.1 LFAST clocking

The MPC5744P includes an LFAST module to support high speed device communications. A single LFAST PLL, which requires a 10-20 MHz reference, supports high speed operation of the LFAST module. For low speed LFAST operation, the reference clock is used directly by the LFAST module.

The LFAST PLL requires a 20 MHz reference. The source may be the PLL0_PHI or the external oscillator (XOSC) or input from the external LFAST device via the LFAST_REF_CLK package pin. When this reference clock is generated internally, it can also be output on the LFAST_REF_CLK pin. The LFAST_REF_CLK pin connection appears in the following figure. Input/Output direction of the LFAST_REF_CLK pin is controlled by the MSCR in the SIUL2.

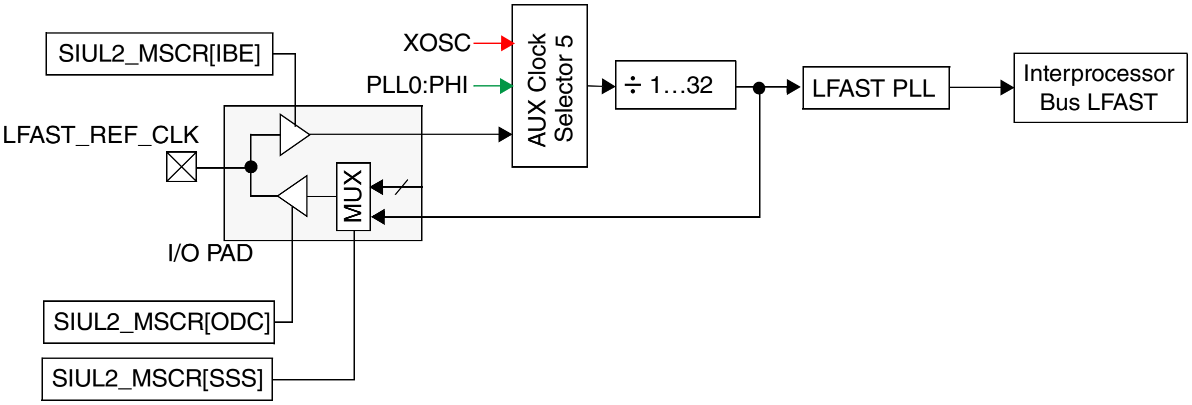


Figure 13-4. Device LFAST SysClk

13.6.2 FlexRay clocking

The FlexRay protocol clock is sourced from either the non-FM PLL0_PHI clock or the external oscillator (XOSC). Select between these sources by programming the FlexRay module's FR_MCR[CLKSEL] field.

- When using the external oscillator clock, a 40 MHz crystal is required for proper operation.
- When using PLL0_PHI, the MC_CGM_AC1_DC0 register must be configured to provide 80 MHz on FRAY_PLL_CLK for proper operation.
    - The frequency of the PLL source must be double the frequency of the XOSC source because the PLL has a less stable duty cycle.
    - The FlexRay module itself divides the PLL frequency to provide a steady, 50% duty cycle, 40 MHz clock internally.
    - This internal FlexRay divider is driven automatically when FR_MCR[CLKSEL] is 1. The user cannot change this internal divider or select it for the XOSC source.

13.6.3 FlexCAN clocking

The CAN_CLK is generated from either the XOSC or PLL0_PHI clock. The PLL0_PHI clock can be divided by a value written to the CGM_AC2_DC register in the MC_CGM. The output of this divider (CAN_PLL_CLK) drives one of two inputs of a multiplexer. The other input is driven by XOSC. FlexCAN_CTRL1[CLK_SRC] selects whether CAN_PLL_CLK or XOSC clock drives the output of the multiplexer for the CAN_CLK.

13.7 Clock monitoring

For all safety critical clocks, the microcontroller detects a missing clock or incorrect frequency. To achieve this clock monitoring units (CMUs) are used. Each CMU is programmed independently. The IRCOSC is used as the clock monitor references. Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

13.7.1 CMU configuration

This section explains the CMU configuration.

Figure 13-5 shows the block diagram for CMU0 on the MPC5744P.

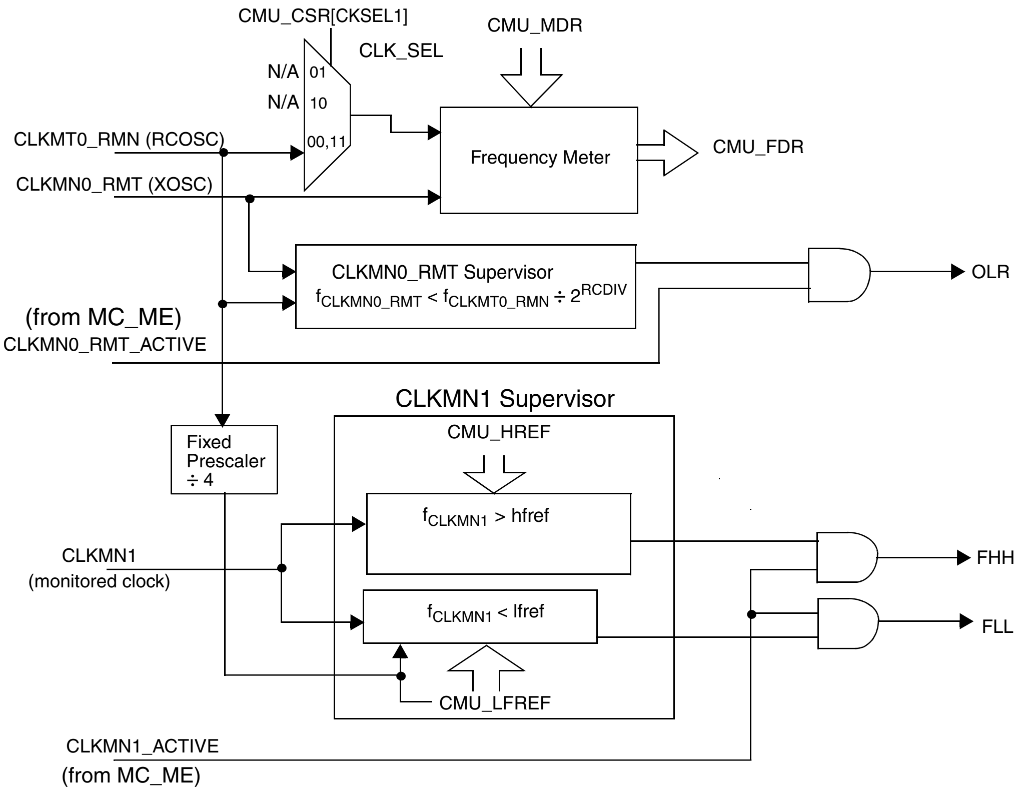


Figure 13-5. CMU0 Block Diagram

Figure 13-6 shows the block diagram for CMU[1:4] on the MPC5744P.

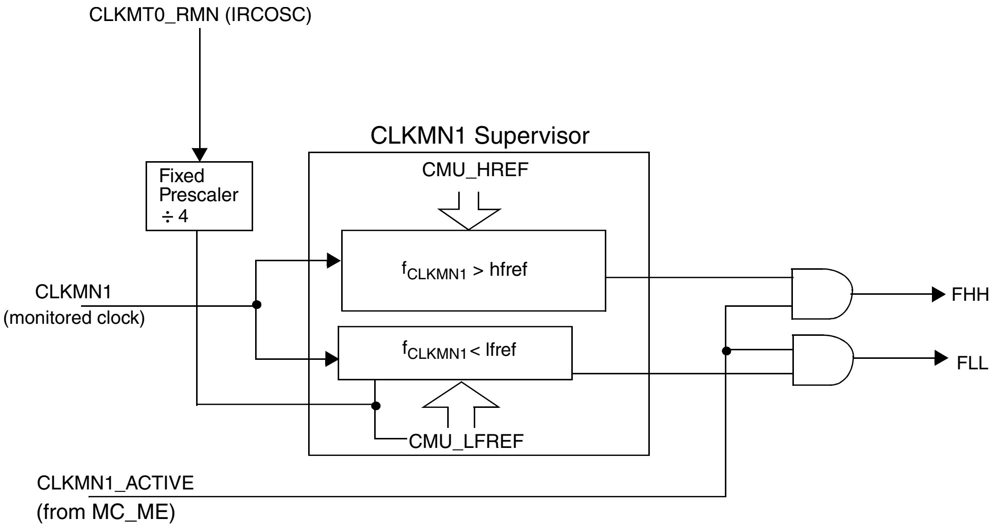


Figure 13-6. CMU[1:4] Block Diagram

13.7.1.1 Clock input sources

Table 13-6 shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRCOSC is the reference clock for all clock monitors. Only CMU0 implements the XOSC monitor. CMU0 uses the IRCOSC clock to measure if the XOSC is too low. CMU0 can also be used to calibrate the IRCOSC frequency using the XOSC. All other CMUs are configured identically.

Table 13-6. Clock input sources

Clock module	Monitored clock
CMU0	MOTC_CLK (CLKMN1 for CMU0), XOSC, IRCOSC
CMU1	CHKR_CLK
CMU2	PBRIDGE0_CLK, PBRIDGE1_CLK
CMU3	ADC_CLK
CMU4	SENT_CLK

13.7.1.2 CMU registers and field availability

Table 13-7 specifies which registers and fields are available on a given CMU for this device.

Table 13-7. CMU registers availability

Address Offset	Register	CMU	Note
0x0000	CMU_CSR	All	CMU_CSR[SFM] and CMU_CSR[CKSEL1] apply in CMU0 only. For all CMU instances on this chip, CMU_CSR[RCDIV] resets to 11b.
0x0004	CMU_FDR	CMU0	—
0x0008	CMU_HFREFR	All	—
0x000C	CMU_LFREFR	All	—
0x0010	CMU_ISR	All	CMU_ISR[OLRI] is in CMU0 only.
0x0014	Reserved	—	—
0x0018	CMU_MDR	CMU0	—

13.7.2 External oscillator (XOSC) monitor

The XOSC frequency is compared to a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set, and an interrupt is generated, if enabled. Frequency limits are give in the MPC5744P Data Sheet.

13.7.3 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

13.7.4 System clock monitors

A CMU is assigned to monitor the frequency of the checker core, both Peripheral bridges, motor control, ADC and SENT clocks (see Figure 13-1).

13.7.4.1 Monitor control and behavior

Software can program an upper and lower limit of the expected clock frequency. If the monitor is enabled and the measured frequency is above or below the limits, a flag bit is set, and an interrupt will be generated, if enabled. The default condition of the clock monitor is disabled.

13.8 Loss of clock sources behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and it provides several options for reaction to a loss of clock in the application. Figure 13-7 provides a high-level view of the logic of the loss of clock sources.

13.8.1 Loss of PLL/XOSC clock

As shown in Figure 13-7, each loss of lock signal from the PLL and XOSC failure signals are monitored by the FCCU. For the event when a clock failure occurs, the FCCU can be programmed to generate a short or long reset sequence or simply to generate an interrupt. In the case of the interrupt, only the unpredictable free running clock is still provided by the PLL. There is no automatic system clock switch in this case, and the user is required to program the switch through the Mode Entry module (MC_ME). Therefore, it is strongly recommended that the interrupt reaction to a PLL loss of lock is not selected when the PLL is used as the source of the system clock.

If a long or short reset is selected: the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states, and the system clock is switched to the IRCOSC.

13.8.2 Loss of IRCOSC clock

The frequency of the IRCOSC clock is monitored by the frequency meter in CMU_0. There is no automated trigger of an FCCU error condition if the IRCOSC fails. Since the IRCOSC is the boot and backup clock, a failure is a catastrophic failure.

Each CMU's FHH and FLL event indicators are connected solely to the FCCU. In this regard, CMU_0's connection to the FCCU in the following figure represents all CMU instances.

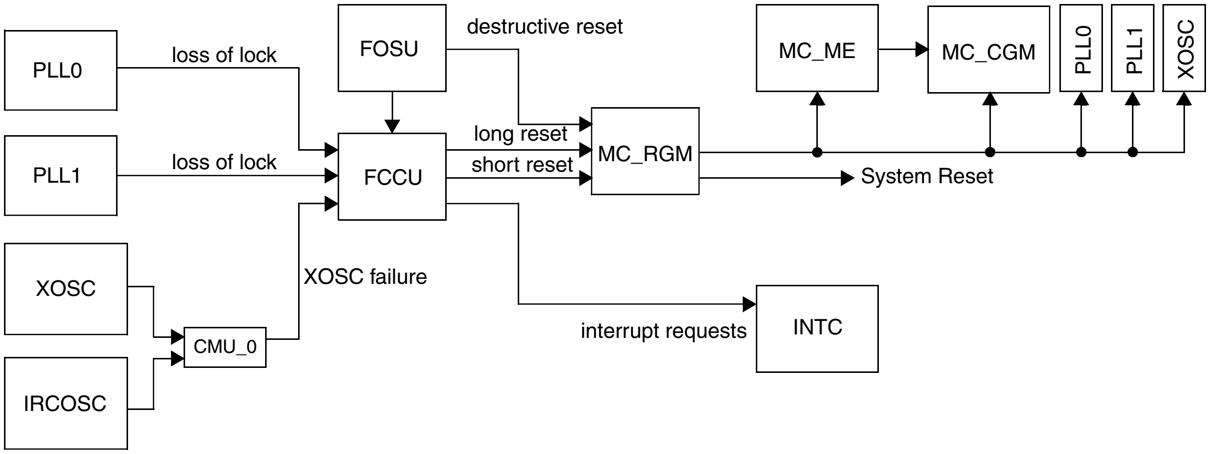


Figure 13-7. Loss of clock sources

13.8.3 PLL loss of lock interrupt

Always write 1 to both PLL0CR[LOLIE] and PLL1CR[LOLIE] to keep the loss of lock interrupt function enabled.

The main interrupt functionality is determined by the FCCU via fault input NCF[24] for PLL0 and fault input NCF[25] for PLL1.

Chapter 14
e200z4d Core Complex Overview

14.1 Introduction

The main computational shell consists of Main Core_0, which uses an e200z4251n3 core. A second core, referred to as Checker Core_0s, is an e200z424 core that is a true subset of the e200z4251n3 core. When enabled, Checker Core_0s operates in lock step mode with Main Core_0, executing exactly the same instructions as Main Core_0. Thus, Checker Core_0s checks to insure that Main Core_0 is executing correctly. This chapter provides overviews of the e200z4251n3 core and the e200z424 core.

The e200z4251n3 and the e200z424 cores are very similar. The register set for these cores is identical. The e200z424 core differs in hardware from the e200z4251n3 in that no Instruction Cache, Data Cache, or D-memory is implemented.

14.2 Overview of e200z4251n3 and e200z424 cores

The e200z processor family is a set of CPU cores that implement low-cost versions of the Power architecture.

The e200z4251n3 and e200z424 are dual-issue, 32-bit Power VLE compliant designs with 32-bit general purpose registers (GPRs).

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision embedded numeric operations using the general-purpose registers.

The e200z4251n3 and e200z424 cores implement the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in PowerISA 2.06, a separate document. The base *PowerISA 2.06* fixed-length 32-bit instruction set is not directly supported.

The e200z4251n3 and e200z424 processor cores integrate a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

A Nexus Class 3+ module is integrated into each e200z4251n3 and e200z424 processor core.

A Memory Protection Unit is integrated in each e200z4251n3 and e200z424 processor core.

14.3 Features

The following is a list of some of the key features of the e200z4251n3 and e200z424 processor cores:

- Dual issue, 32-bit *PowerISA 2.06*-VLE compliant CPU
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - Dedicated branch address calculation adder
  - Branch target prefetching using BTB
- Load/store unit
  - 2 cycle load latency
  - Fully pipelined
  - Misaligned access support
- 32-bit General Purpose Register (GPR) file
- Dual AHB 2.v6 64-bit System buses
- Local Data Memory (DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z4251n3 core: 64 KB DMEM
  - e200z424 core does not contain either Instruction or Data MEM
- Memory Protection Unit (MPU) implementing a 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized instruction or data memory regions
- 2-Way Set Associative Harvard Instruction and Data Cache
  - e200z4251n3 core: 8 KB ICache, 4 KB DCache
  - e200z424 core does not contain either Instruction or Data Cache
- Embedded Floating-point APU (EFPU2) supporting single-precision floating-point operations
- Lightweight Signal Processing Extension (LSP) APU to support real-time SIMD fixed-point embedded numerics operations using the GPRs
- Performance Monitor APU supporting execution profiling

- Nexus Class 3+ Real-time Development Unit
- Power management
    - Low power design with extensive clock gating
    - Power saving modes: halt, stop, wait
    - Dynamic power management of execution units, caches and Local memories
- Testability
    - Synthesizeable, MuxD scan design
    - ABIST/MBIST for arrays
    - Built-in LBIST unit

14.4 Microarchitecture summary

The e200z4251n3 and e200z424 processor cores utilize a five stage instruction pipeline with two stages for execution. The Instruction Fetch, Instruction Decode/Register File Read/EA Calc, Execute0/Memory Access0, Execute1/Memory Access1, and Register Writeback stages operate in an overlapped fashion allowing single clock instruction execution for most instructions. The following figure shows a block diagram of the e200z architecture.

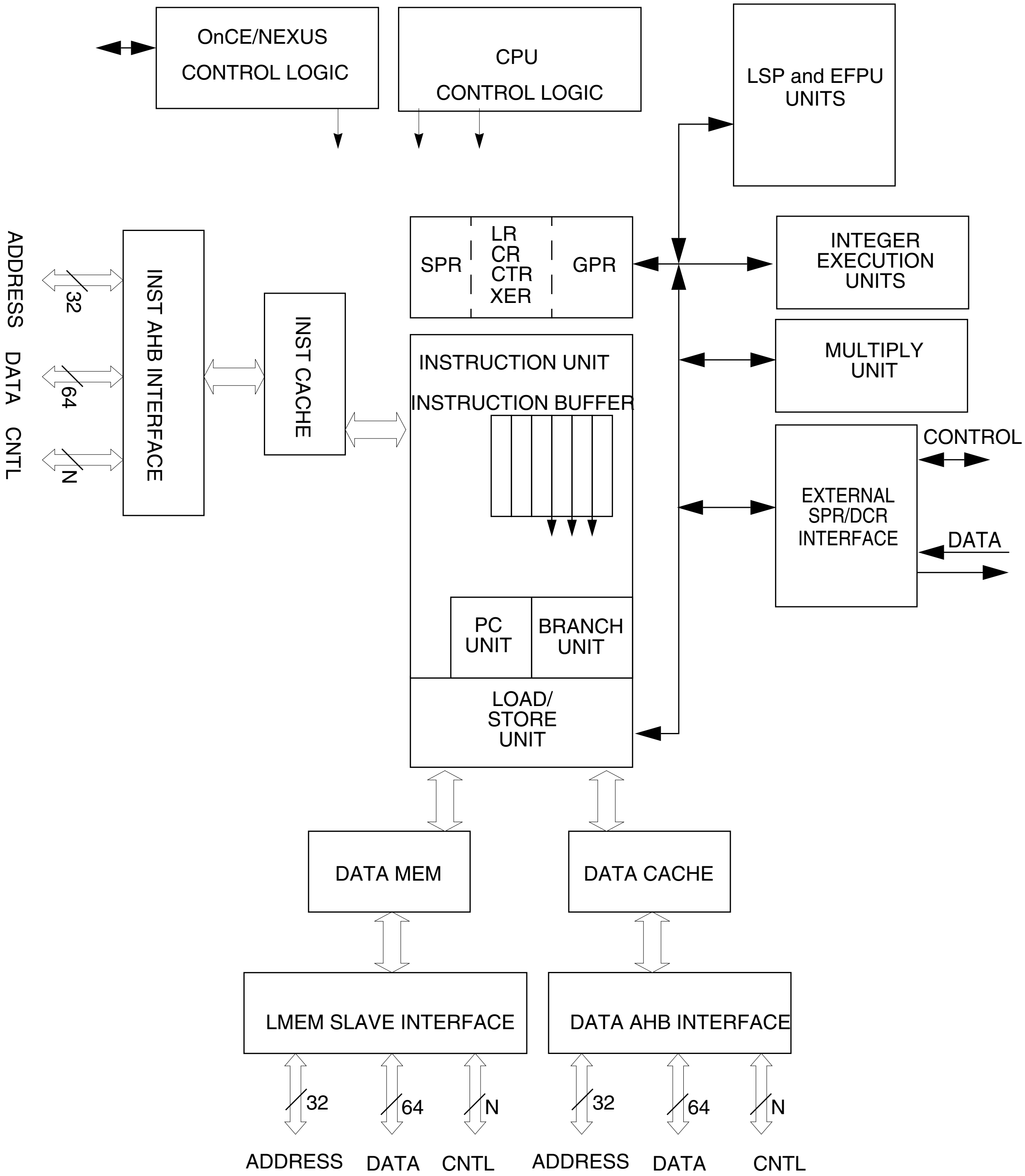


Figure 14-1. e200z block diagram

The integer execution units each consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), and result feed-forward hardware. Integer EU1 also supports hardware division and a 32x32 Hardware Multiplier array.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a pipelined hardware array, and the divide instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

The Instruction Unit contains a PC incrementer and dedicated Branch Address adders to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in an execution time of two clocks for correctly predicted branches. Conditional branches which are not taken execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, word (32-bit), and doubleword data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations and use 64-bit data transfers when possible. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

The Condition Register unit supports the condition register (CR) and condition register operations defined by the Power architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The LSP APU supports vector instructions operating on 16- and 32-bit fixed-point data types. The EFPU2 APU supports 32-bit IEEE-754 single-precision floating-point formats, supports scalar single-precision floating-point operations, and operates in a pipelined fashion. The general purpose register file is used for source and destination operands, and a unified storage model is used for scalar single-precision floating-point data types of 32-bits and the normal integer type. Low latency fixed-point and floating-point add, subtract, mixed add/subtract, sum, diff, min, max, multiply, multiply-add, multiply-sub, divide, square root, compare, and conversion operations are provided, and most operations can be pipelined.

14.4.1 Instruction unit features

The features of the e200z4251n3 and e200z424 Instruction units are:

- 64-bit path to cache supports fetching of two 32-bit instructions per clock
- Instruction buffer holds up to eight 32-bit instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder and branch lookahead logic (BTB) supporting single cycle execution of successfully predicted branches

14.4.2 Integer unit features

The e200z4251n3 and e200z424 integer units support single cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for the count leading zero's function
- 32-bit single cycle barrel shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 4-14 clocks with minimized execution timing (EU1 only)
- Pipelined 32x32 hardware multiplier array supports 32x32->32 multiply with 2 clock latency, 1 clock throughput (EU1 only)

14.4.3 Load/Store unit features

The e200z4251n3 and e200z424 load/store units support load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions and context save/restore instructions

14.4.4 EFPU2 Floating-Point Unit features

The EFPU2 embedded floating-point unit supports pipelined operation of most floating-point operations, allowing a throughput of one FP operation per cycle, and supports a variety of operations:

- Supports IEEE754 single-precision data format
- Supports IEEE754 half-precision format for conversion <-> single-precision format

- EFPU2 instructions utilize the 32-bit GPRs for minimized context save/restore overhead
- Provides low latency pipelined scalar floating-point add, subtract, add/sub, sub/add, multiply, multiply-add, min, max, absolute value, compare, and conversion instructions with single-cycle throughput
- Default results mode for real-time applications allows for exception-free operations when underflow or overflows occur
- Trap Enable controls allow for trapping of various operation exceptions and emulation of IEEE754 boundary case results
- Floating-point Divide and Square root logic operating in 13 (FP div) and 11 (FP sqrt) clocks

14.4.5 LSP Lightweight Signal Processing Unit features

The LSP APU is designed to accelerate signal processing applications normally suited to DSP operation.

- LSP instructions operate on the 32-bit GPRs
- Supports SIMD integer operations: arithmetic, logical, shift
- Supports SIMD rounding, saturation, pack, unpack, merge, extract, select for efficient data manipulation
- Supports SIMD multiply/multiply accumulate/ dot product operations
- 16-, 32-, and (limited) 64-bit signed and unsigned integer data types
- 16- and 32-bit signed fractional support (1.15, 1.31 representations)
- Guarded 32-bit and 64-bit signed fractional support (9.23, 33.31, and 17.47 representations)
- Single-cycle throughput for all instructions (except frac divide)
- Supports fractional divide operations (32/32->32)
- Specialized operations such as field manipulation and count-leading
- Supports specialized load/store operations with integrated data manipulation
- Multiple addressing modes supported including circular and bit-reversed addressing
- Sustained throughput of up to 2 16x16-bit multiply/accumulate operations per cycle.

14.4.6 MPU features

The features of the MPU are as follows:

- 24-entry fully-associative Range Table
- Arbitrary range size support from 1 byte to 4 GB
- 8-bit Process Identifier
- Bypass capability for individual access types
- Maskable Address and Process Identifier

- Programmable Memory Attributes
- Entry Invalidation Protection
- Write-Once Option
- Alternate Debug Breakpoint/Watchpoint function

14.4.7 Cache features

The features of the caches are as follows:

- 2-Way Set Associative Harvard Instruction and Data Caches
  - e200z4251n3: 8 KB ICache, 4 KB DCache
  - e200z424: no ICache, no DCache
- Writethrough Support
- 8-entry Store Buffer
- Linefill Buffer
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and 64-bit write data bus
- Support for Way locking
- Support for Write allocation policies
- Support for multi-bit EDC with Correction/Auto-invalidation capability for the I and D caches
- Hardware Debug Cache Invalidate Support for the Data Cache
- Software and Hardware Debug access to cache tag, status and data storage via dedicated control registers

14.4.8 Local memory features

The features of the local data memories are as follows:

- Local data memories (DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z4251n3: 64 KB DMEM
  - e200z424: no IMEM, no DMEM
- 1 wait-state slave DMEM read access, 1 wait-state slave DMEM write access
- 64-bit local bus to CPU supporting 0 wait-state CPU access
- Support for multi-bit (SECDED) ECC with correction/scrubbing capability
- 4-entry ECC R-M-W Store Buffer for DMEM
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and (for DMEM only) 64-bit write data bus

14.4.9 Performance Monitor Unit features

The features of the performance monitor include:

- Four configurable 32-bit performance monitor counters, each capable of counting selected CPU subsystem events
- Performance Monitor interrupt
- Ability to configure performance monitor resources for debugger use
- Hardware input signals for qualification of counting by individual counters
- Hardware output signals to indicate counter overflows
- Dedicated watchpoint outputs for each counter with programmable periodicity
- Trigger On/Off control for each counter based on subsystem events
- Multiple selectable event types including # of clocks, CPU stall cycles, # of instructions completed, # of interrupts taken, # of memory accesses (by type), # of instruction or data cache misses, # of cycles with 0, 1, or 2 instructions issued, # of instructions in a given class completed (ldst, branch, integer, FP, and so on), and cache linefills

14.4.10 e200z4251n3 system bus features

The features of the e200z4251n3 system bus interface include:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture (AMBA) AHB2.v6 protocol
- 32-bit address buses, 64-bit instruction-side and data-side system buses
- Data interface provides separate uni-directional read and write data buses
- Support for HCLK running at a slower rate than CPU clock

14.4.11 e200z424 system features

Checker Core_0 is implemented using an e200z424 core. When enabled, Checker Core_0s is used in lock step with Main_Core_0. The e200z424 core can only be enabled or disabled by the user. For this device, the Checker Core_0s has no system bus and cannot independently execute user code. The core is only used to execute the same instruction stream as Main Core 0.

The e200z424 core is identical to the e200z4251n3 with five exceptions:

- IMEM memory array is not present
- ICache memory array is not present
- DMEM memory array is not present
- DCache memory array is not present
- e200z424 has a pair of system bus interfaces, which are driven/sampled by the delayed lock step logic

Chapter 15
Core Detailed Description

15.1 Overview of the e200z4251n3 core

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The e200z4251n3 is a dual-issue 32-bit *Power ISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z4251n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *Power ISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

A Lightweight Signal Processing Extension (LSP) APU is provided to support real-time SIMD fixed-point embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). The GPRs support register pairing in order to support vector instructions defined by the LSP APU. These instructions operate on a vector pair of 16-bit or 32-bit data types, and deliver 16-bit vector and 32-bit scalar results. A wide variety of data manipulation, multiply, multiply/accumulate, and dot product instructions along with specialized memory access instructions allow a sustained throughput of up to two 16x16-bit multiply/accumulate operations per cycle.

The e200z4251n3 core integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The e200z4251n3 core contains an 8 KB Instruction Cache, and a 4 KB Data Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

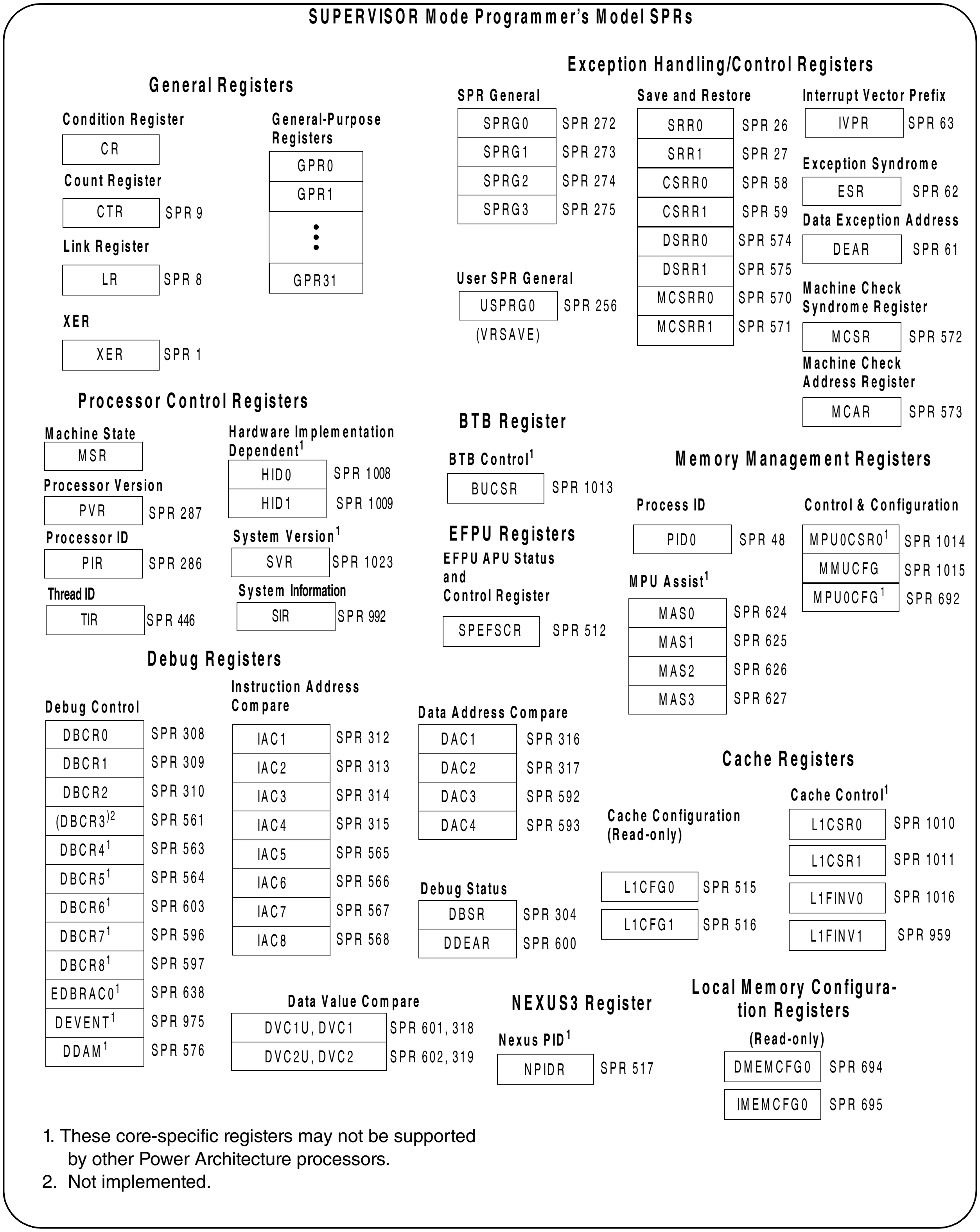
A Memory Protection Unit that protects various instruction and data memory areas is also included.

15.2 Register model

This section describes the registers implemented in the e200z4251n3 core. The *Power ISA 2.06* architecture defines register-to-register operations for all computational instructions.

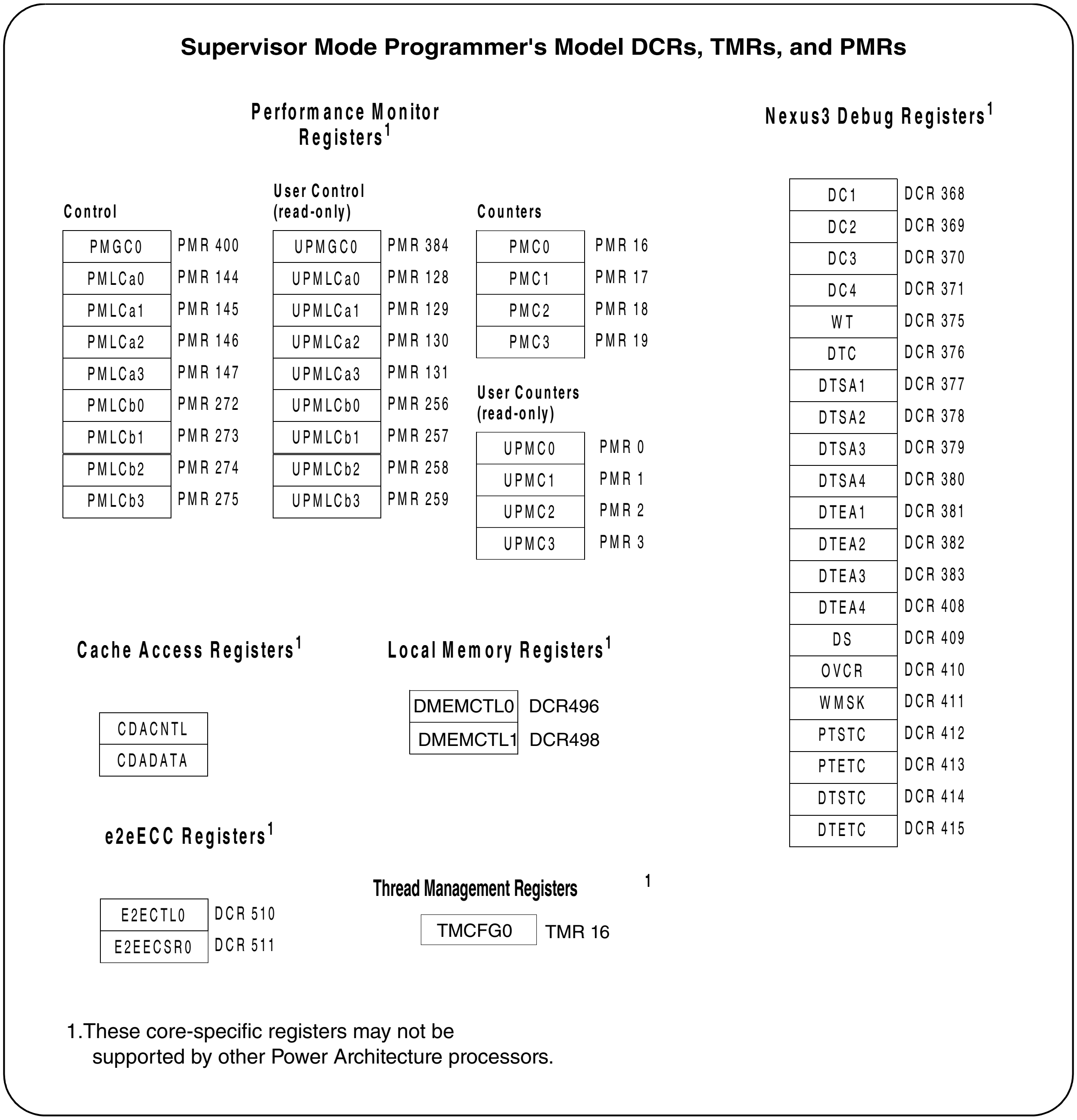
e200z4251n3 extends usage of the general purpose registers to support EFPU /LSP APU operations on the 32-bit GPR registers.

Figure 15-1 and Figure 15-2 show the complete e200z4251n3 register set, all of which are accessible in supervisor mode. Figure 15-3 and Figure 15-4 show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).



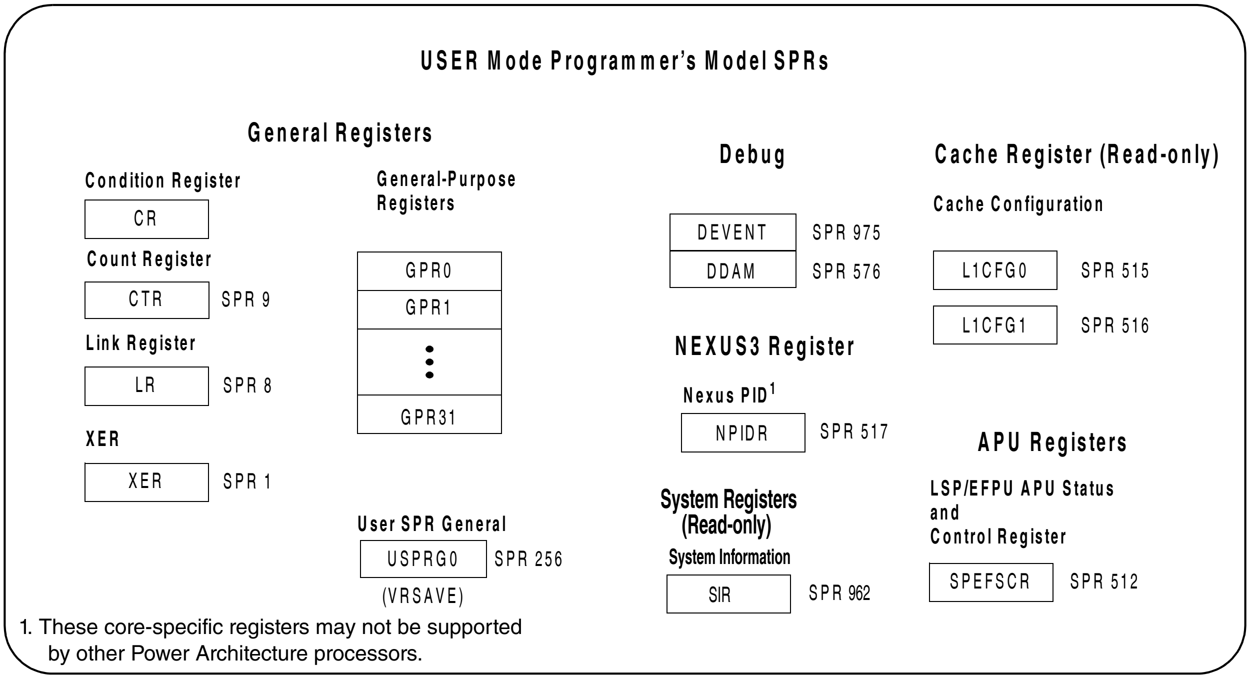
1. These core-specific registers may not be supported by other Power Architecture processors.
2. Not implemented.

Figure 15-1. e200z4251n3 Supervisor Mode Programmer's Model SPRs



1.These core-specific registers may not be supported by other Power Architecture processors.

Figure 15-2. e200z4251n3 Supervisor Mode Programmer's Model DCRs and PMRs



1. These core-specific registers may not be supported by other Power Architecture processors.

Figure 15-3. e200z4251n3 User Mode Programmer's Model SPRs

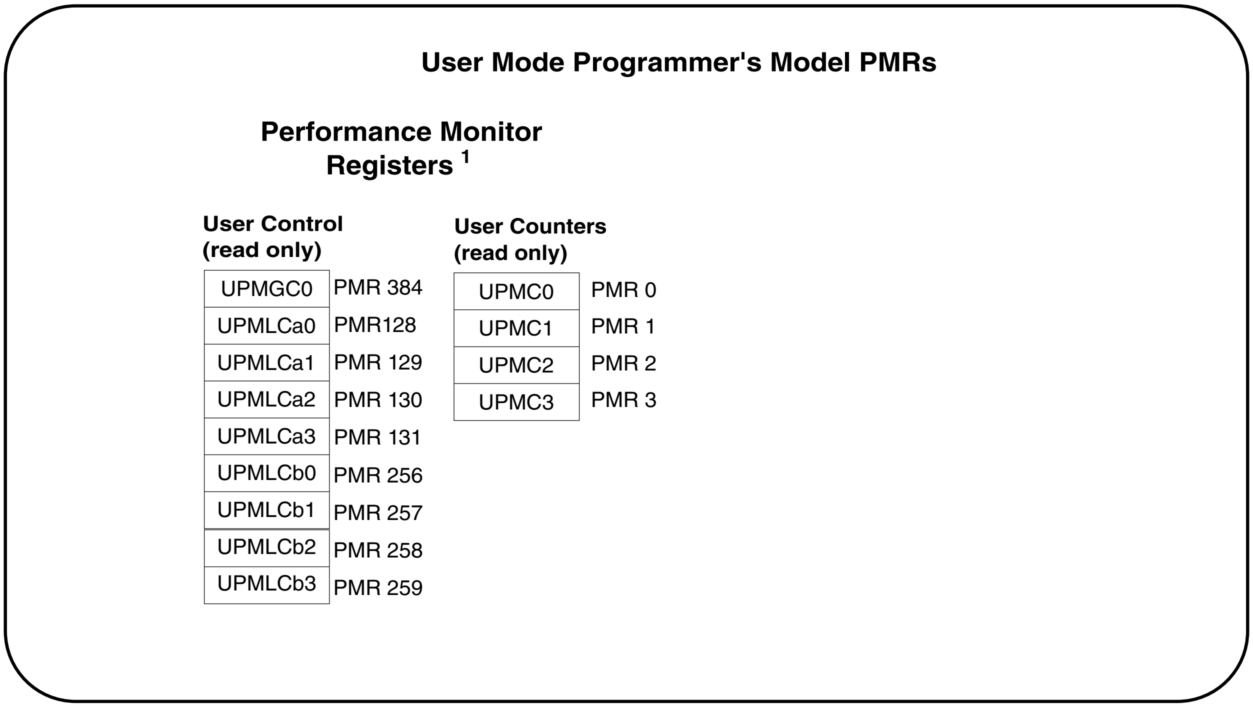


Figure 15-4. e200z4251n3 User Mode Programmer's Model PMRs

15.2.1 Branch Unit Control and Status Register (BUCSR)

The BUCSR register is used for general control and status of the branch target buffer (BTB). The BTB is detailed in . The BUCSR is shown in the following figure.

<table>
<tr><td colspan="22">0</td><td>BBFI</td><td colspan="3">0</td><td colspan="2">BALLOC</td><td>0</td><td colspan="2">BPRED</td><td>BPEN</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
</table>

SPR — 1013; Read/Write; Reset — 0x0

Figure 15-5. Branch Unit Control and Status Register (BUCSR)

The BUCSR fields are defined in the following table.

Table 15-1. Branch Unit Control and Status Register (BUCSR)

Bits	Name	Description
0:21 [32:53]	-	Reserved
22 [54]	BBFI	Branch target buffer flash invalidate. When written to a '1', BBFI flash clears the valid bit of all entries in the branch target buffer; clearing occurs regardless of the value of the enable bit (BPEN). Note: BBFI is always read as 0.
23:25 [55:57]	-	Reserved
26:27 [58:59]	BALLOC	Branch Target Buffer Allocation Control. This field controls BTB allocation for branch acceleration when BPEN = 1. Note that BTB hits are not affected by the settings of this field. 00 Branch Target Buffer allocation for all branches is enabled. 01 Branch Target Buffer allocation is disabled for backward branches. 10 Branch Target Buffer allocation is disabled for forward branches. 11 Branch Target Buffer allocation is disabled for both branch directions.
28 [60]	-	Reserved
29:30 [61:62]	BPRED	Branch Prediction Control (Static). This field controls operation of static prediction mechanism on a BTB miss. When a branch is predicted as taken, fetching of the predicted target location will be performed for branch acceleration. BPRED operates independently of BPEN, and with a BPEN setting of 0, will be used to perform static prediction of all unresolved branches. 00 Branch predicted taken on BTB miss for all branches. 01 Branch predicted taken on BTB miss only for forward branches. 10 Branch predicted taken on BTB miss only for backward branches. 11 Branch predicted not taken on BTB miss for both branch directions. **NOTE:** BPRED functionality is not supported in e200z4xxx processor cores due to its short pipe stages.
31 [63]	BPEN	Branch target buffer prediction enable. When the BPEN bit is cleared, no hits will be generated from the BTB, and no new entries will be allocated. Entries are not automatically invalidated when BPEN is cleared; the BBFI bit controls entry invalidation. BPEN operates independently of BPRED, and will be used even with a BPRED setting of 00.

Table 15-1. Branch Unit Control and Status Register (BUCSR)

Bits	Name	Description
		0 Branch target buffer prediction disabled 1 Branch target buffer prediction enabled (enables BTB to predict branches)

15.3 Dual-issue operation

The instruction issue unit attempts to issue a pair of instructions to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. As many as three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

The following table shows the concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue of those instructions.

Table 15-2. Concurrent instruction issue capabilities

Class of instruction	Branch	Load/Store	Scalar integer	Scalar float	Special
Branch	—	yes	yes	yes	—
Load/Store	yes	—	yes	yes	—
Scalar integer	yes	yes	yes[1]	yes	—
Scalar float	yes	yes	yes[2]	—	—
Special	—	—	—	—	—

[1] Excludes multiply or divide class instructions occurring in both issue slots.

[2] Excludes divide/sqrt class instructions occurring in both issue slots

15.4 Instruction timing

Instruction timing in number of processor clock cycles for various instruction classes is shown in Table 15-3. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during execution.

Timing for EFPU instructions is detailed in Table 15-6.

Timing for LSP instructions is detailed in:

- LSP simple instruction timing—Table 15-7
- LSP complex instruction timing—Table 15-8
- LSP shift/rotate instruction timing—Table 15-9
- LSP vector compare instruction timing—Table 15-10
- LSP vector select instruction timing—Table 15-11
- LSP vector data arrangement instruction timing—Table 15-12
- LSP multiply and multiply/accumulate instruction timing—Table 15-13
- LSP dot product instruction timing—Table 15-14
- LSP miscellaneous vector instruction timing—Table 15-15
- LSP load and store instruction timing—Table 15-16

Load/store multiple instruction cycles are represented as a fixed number of cycles plus a variable number of cycles where *n* is the number of words accessed by the instruction. In addition, cycle times marked with & require variable number of additional cycles due to serialization.

Table 15-3. Instruction class cycle counts

Class of Instructions	Latency	Throughput	Special notes
integer: **add**, **sub**, **shift**, **rotate**, **logical**, **cntlzw**	1	1	—
integer: **compare**	1	1	—
Branch	3/2/1	3/2/1	Correct branch lookahead allows single-cycle execution. Worst-case mispredicted branch is 3 cycles.
multiply	2	1	—
divide	4–14	4–14	Data-dependent timing
CR logical	1	1	—
loads (non-multiple)	2	1	—
load multiple	2 + *n*/2 (max)	1 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
stores (non-multiple)	2	1	—

Table continues on the next page...

Table 15-3. Instruction class cycle counts (continued)

Class of Instructions	Latency	Throughput	Special notes
store multiple	2 + *n*/2 (max)	1 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
mtmsr, **wrtee**, **wrteei**	3&	3&	—
mcrf	1	1	—
mfspr, **mtspr**	4&	4&	Applies to Debug SPRs, optional unit SPRs
mfspr, **mfmsr**	1	1	Applies to certain limited internal, non Debug SPRs
mfcr, **mtcr**	1	1	—
se_rfi, **se_rfci**, **se_rfdi**, **se_rfmci**	3	—	—
se_sc, **e_sc**	4	—	—
e_tw	4	—	Trap taken timing

Detailed timing for each instruction mnemonic along with serialization requirements is shown in Table 15-4 and Table 15-5.

Table 15-4. Instruction timing by mnemonic—16-bit instructions

Mnemonic	Latency	Serialization
se_add	1	—
se_addi	1	—
se_and[.]	1	—
se_andc	1	—
se_andi	1	—
se_bc	3/2/1	—
se_bclri	1	—
se_bctr	3/2	—
se_bctrl	3/2	—
se_bgeni	1	—
se_bl	3/2/1	—
se_blr	3/2	—
se_blrl	3/2	—
se_bmaski	1	—
se_b	3/2/1	—
se_bseti	1	—
se_btsti	1	—
se_cmp	1	—
se_cmph	1	—
se_cmphl	1	—
se_cmpi	1	—

Table continues on the next page...

Table 15-4. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_cmpl	1	—
se_cmpli	1	—
se_dnh	1+	—
se_dni	1+	—
se_extsb	1	—
se_extsh	1	—
se_extzb	1	—
se_extzh	1	—
se_illegal	4	—
se_isync	6[1]	Refetch
se_lbz	2	—
se_lhz	2	—
se_li	1	—
se_lwz	2[2]	—
se_mfar	1	—
se_mfctr	1	—
se_mflr	1	—
se_mr	1	—
se_mtar	1	—
se_mtctr	1	—
se_mtlr	1	—
se_mullw	2	—
se_neg	1	—
se_not	1	—
se_or	1	—
se_rfci	3	Refetch
se_rfdi	3	Refetch
se_rfi	3	Refetch
se_rfmci	3	Refetch
se_sc	4	Refetch
se_slw	1	—
se_slwi	1	—
se_sraw	1	—
se_srawi	1	—
se_srw	1	—
se_srwi	1	—
se_stb	2	—
se_sth	2[2]	—
se_stw	2[2]	—

Table continues on the next page...

Table 15-4. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_sub	1	—
se_subf	1	—
se_subi[.]	1	—

1. Plus additional synchronization time.
2. Aligned

Table 15-5. Instruction timing by mnemonic—32-bit instructions

Mnemonic	Latency	Serialization
add[.]	1	—
e_add16i	1	—
e_add2i.	1	—
e_add2is	1	—
addc[.]	1	—
addco[.]	1	—
adde[.]	1	—
addeo[.]	1	—
e_addi[.]	1	—
e_addic[.]	1	—
addme[.]	1	—
addmeo[.]	1	—
addo[.]	1	—
addze[.]	1	—
addzeo[.]	1	—
and[.]	1	—
e_and2i.	1	—
e_and2is.	1	—
andc[.]	1	—
e_andi[.]	1	—
e_b	3/2/1	—
e_bc	3/2/1	—
e_bcl	3/2/1	—
e_bl	3/2/1	—
cmp	1	—
e_cmp16i	1	—
e_cmph	1	—
e_cmph16i	1	—
e_cmphl	1	—
e_cmphl16i	1	—
e_cmpi	1	—

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
cmpl	1	—
e_cmpl16i	1	—
e_cmpli	1	—
cntlzw[.]	1	—
e_crand	1	—
e_crandc	1	—
e_creqv	1	—
e_crnand	1	—
e_crnor	1	—
e_cror	1	—
e_crorc	1	—
e_crxor	1	—
dcba	1	—
dcbf	1	—
dcbi	1	—
dcbst	1	—
dcbt	1	—
dcbtst	1	—
dcbz	— (alignment error)	—
divw[.]	4–14	—
divwo[.]	4–14[1]	—
divwu[.]	4–14[1]	—
divwuo[.]	4–14[1]	—
e_dnh	1+	—
e_dni	1+	—
eqv[.]	1	—
extsb[.]	1	—
extsh[.]	1	—
icbi	1	—
icbt	1	—
isel	1	—
lbarx	2	—
e_lbz	2	—
e_lbzu	2	—
lbzux	2	—
lbzx	2	—
e_lha	2	—
lharx	2	—
e_lhau	2[2]	—

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
lhaux	2[2]	—
lhax	2[2]	—
lhbrx	2[2]	—
e_lhz	2[2]	—
e_lhzu	2[2]	—
lhzux	2[2]	—
lhzx	2[2]	—
e_li	1	—
e_lis	1	—
e_lmw	2 + (*n*/2)	—
lwarx	2	—
lwbrx	2[2]	—
e_lwz	2[2]	—
e_lwzu	2[2]	—
lwzux	2[2]	—
lwzx	2[2]	—
mbar	1	Pseudo-dispatch
e_mcrf	1	—
mcrxr	1	Completion
mfcr	1	—
mfdcr	4[3]	Completion
mfmsr	1	—
mfspr (except DEBUG, CACHE, LMEM < MPU)	1	None
mfspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mpure	4[3]	Completion
mpusync	1	Completion
mpuwe	4[3]	Completion
msync	1[3]	Completion
mtcrf	2	—
mtmsr	3[3]	Completion
mtspr (except DEBUG, **msr**, hid0/1)	1	None
mtspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mulhw[.]	2	—
mulhwu[.]	2	—
e_mull2i	2	—
e_mulli	2	—
mullw[.]	2	—

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
mullwo[.]	2	—
nand[.]	1	—
neg[.]	1	—
nego[.]	1	—
nor[.]	1	—
or[.]	1	—
e_or2i	1	—
e_or2is	1	—
orc[.]	1	—
e_ori[.]	1	—
e_rlw[.]	1	—
e_rlwi[.]	1	—
e_rlwimi	1	—
e_rlwinm	1	—
e_sc	4	Refetch
slw[.]	1	—
e_slwi[.]	1	—
sraw[.]	1	—
srawi[.]	1	—
srw[.]	1	—
e_srwi[.]	1	—
e_stb	2	—
stbcx.	2	—
e_stbu	2	—
stbux	2	—
stbx	2	—
e_sth	2[2]	—
sthbrx	2[2]	—
sthcx.	2	—
e_sthu	2[2]	—
sthux	2[2]	—
sthx	2[2]	—
e_stmw	2 + (*n*/2)	—
e_stw	2[2]	—
stwbrx	2[2]	—
stwcx.	2	—
e_stwu	2[2]	—
stwux	2[2]	—
stwx	2[2]	—

Table continues on the next page...

Table 15-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
subf[.]	1	—
subfc[.]	1	—
subfco[.]	1	—
subfe[.]	1	—
subfeo[.]	1	—
e_subfic[.]	1	—
subfme[.]	1	—
subfmeo[.]	1	—
subfo[.]	1	—
subfze[.]	1	—
subfzeo[.]	1	—
tw	4	—
wait	—	Completion
wrtee	3	Completion
wrteei	3	Completion
xor[.]	1	—
e_xori[.]	1	—

1. With early-out capability, timing is data-dependent.
2. Aligned.
3. Plus additional synchronization time.

Instruction timing for EFPU single-precision scalar floating-point instructions is shown in Table 15-6. The table is sorted by opcode.

Table 15-6. EFPU single-precision scalar floating-point instruction timing

Instruction	Latency	Throughput	Comments
efsabs	2	1	—
efsadd	2	1	—
efscfh	2	1	—
efscfsf	2	1	—
efscfsi	2	1	—
efscfuf	2	1	—
efscfui	2	1	—
efscmpeq	2	1	—
efscmpgt	2	1	—
efscmplt	2	1	—
efscth	2	1	—
efsctsf	2	1	—
efsctsi	2	1	—
efsctsiz	2	1	—

Table continues on the next page...

Table 15-6. EFPU single-precision scalar floating-point instruction timing (continued)

Instruction	Latency	Throughput	Comments
efsctuf	2	1	—
efsctui	2	1	—
efsctuiz	2	1	—
efsdiv	13	13	Blocking, no execution overlap with next instruction
efsmadd	2	1	Destination also used as source
efsmsub	2	1[1]	Destination also used as source
efsmax	2	1	—
efsmin	2	1	—
efsmul	2	1	—
efsnabs	2	1	—
efsneg	2	1	—
efsnmadd	2	1[1]	Destination also used as source
efsnmsub	2	1[1]	Destination also used as source
efssqrt	11	11	Blocking, no overlap with next instruction
efssub	2	1	—
efststeq	2	1	—
efststgt	2	1	—
efststlt	2	1	—

[1] Destination register is also a source register, so for full throughput, back-to-back operations must use a different destination register.

Instruction timing in number of processor clock cycles for LSP instructions are shown in the following tables. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during divide execution.

Instruction timing for LSP simple instructions is shown in Table 15-7.

Table 15-7. LSP simple instruction timing

Basic operation	Variants	Latency	Throughput
Absolute value	**zvabsh**, **zabsw**	1	1
	zvabshs, **zabsws**	1	1
Add	**zaddd**, **zvaddh**, **zvaddw**	1	1
	zadddss, **zvaddhss**, **zaddwss**, **zvaddwss**, **zadddus**, **zvaddhus**, **zaddwus**	1	1
	zvaddhx, **zvaddhxss**	1	1
	zaddhe[s,u]w, **zaddho[s,u]w**	1	1
	zvaddih	1	1
	zaddwgsf	1	1

Table continues on the next page...

Table 15-7. LSP simple instruction timing (continued)

Basic operation	Variants	Latency	Throughput
	zaddwgsi, **zaddwgui**	1	1
AddSubf	**zvaddsubfh**, **zvaddsubfhss**, **zvaddsubfw**, **zvaddsubfwss**	1	1
	zvaddsubfhx, **zvaddsubfhxss**	1	1
Count Leading	**zvcntlsh**, **zvcntlzh**, **zcntlsw**	1	1
Negate	**zvnegh**	1	1
	zvneghs, **znegws**	1	1
	zvnegho, **zvneghos**	1	1
Round	**zrndwh**, **zrndwhss**	1	1
Saturate	**zsatsdsw**, **zsatsduw**	1	1
	zsatuduw	1	1
	zvsatshuh	1	1
	zvsatuhsh	1	1
	zsatswsh, **zsatswuh**	1	1
	zsatswuw	1	1
	zsatuwsh, **zsatuwuh**	1	1
	zsatuwsw	1	1
Subf	**zsubfd**, **zvsubfh**, **zvsubfw**	1	1
	zsubfdss, **zvsubfhss**, **zsubfwss**, **zvsubfwss**, **zsubfdus**, **zvsubfhus**, **zsubfwus**, **zvsubfwus**	1	1
	zvsubfhx, **zvsubfhxss**	1	1
	zsubfhe[s,u]w, **zsubfho[s,u]w**	1	1
	zvsubifh	1	1
	zsubfwgsf	1	1
	zsubfwgsi, **zsubfwgui**	1	1
SubfAdd	**zvsubfaddh**, **zvsubfaddhss**, **zvsubfaddw**, **zvsubfaddwss**	1	1
	zvsubfaddhx, **zvsubfaddhxss**	1	1

Instruction timing for LSP complex instructions is shown in Table 15-8. For the divide instruction, the number of stall cycles is (latency) for following instructions.

Table 15-8. LSP complex instruction timing

Operation	Instruction	Latency	Throughput
Divide	**zdivwsf**	4 – 14	4 – 14[1]

[1] Timing is data-dependent.

Instruction timing for LSP shift/rotate instructions is shown in Table 15-9.

Table 15-9. LSP shift/rotate instruction timing

Basic operation	Variants	Latency	Throughput
Logical Shift Left	**zvslh**, **zvslhi**	1	1
Logical Shift Right	**zvsrhu**, **zvsrhiu**	1	1
Rotate Left	**zvrlh**, **zvrlhi**	1	1
Signed Shift Left and Saturate	**zslwss**, **zslwiss**, **zvslhss**, **zvslhiss**	1	1
Unsigned Shift Left and Saturate	**zslwus**, **zslwius**, **zvslhus**, **zvslhius**	1	1
Arithmetic Shift Right	**zvsrhs**, **zvsrhis**	1	1

Instruction timing for LSP vector compare instructions is shown in Table 15-10.

Table 15-10. LSP vector compare instruction timing

Basic comparison operation	Instruction	Latency	Throughput
=	**zvcmpeqh**>	1	1
>	**zvcmpgths**, **zvcmpgthu**	1	1
<	**zvcmplthu**, **zvcmplthu**	1	1

Instruction timing for LSP vector select instructions is shown in Table 15-11.

Table 15-11. LSP vector select instruction timing

Operation	Instruction	Latency	Throughput
Select	**zvselh**	1	1

Instruction timing for LSP vector data arrangement instructions is shown in Table 15-12.

Table 15-12. LSP vector data arrangement instruction timing

Operation	Instruction	Latency	Throughput
Merge	**zvmergehih**	1	1
	zvmergehiloh	1	1
	zvmergeloh	1	1
	zvmergelohih	1	1
Pack	**zvpkswshfrs**	1	1
	zvpkswshs, **zvpkswuhs**, **zvpkuwuhs**	1	1
	zpkswgswfrs	1	1
	zvpkshgwshfrs	1	1
Splat	**zvsplatfih**	1	1
	zvsplatih	1	1
Unpack	**zunpkwgsf**	1	1
	zvunpkhgwsf	1	1
	zvunpkhsf, **zvunpkhsi**, **zvunpkhui**	1	1

Instruction timing for LSP multiply and multiply/accumulate instructions is shown in Table 15-13.

Table 15-13. LSP multiply and multiply/accumulate instruction timing

Instruction	Latency	Throughput
all **z[v]m{h,w}** instructions	—	1

Instruction timing for LSP dot product instructions is shown in Table 15-14.

Table 15-14. LSP dot product instruction timing

Instruction	Latency	Throughput
all **z[v]dotp** instructions	—	1

Instruction timing for LSP miscellaneous instructions is shown in Table 15-15.

Table 15-15. LSP miscellaneous vector instruction timing

Operation	Instruction	Latency	Throughput
Circular increment	**zcircinc**	1	1
Bit reversed increment	**zbrminc**	1	1

Instruction timing for LSP load and store instructions instructions is shown in Table 15-16.

Table 15-16. LSP load and store instruction timing

Instruction	Latency	Throughput
all **zv** loads	—	1
all **zv** stores	—	1

15.5 Reservation instructions and cache interactions

If the CPU supports reservation instruction functionality, the CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

15.6 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. This register is also used by the LSP APU. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in the following figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0										FINXS	FINVS	FDBZS	FUNFS	FOVFS	MODE	SOV	OV	FG	FX	FINV	FDBZ	FUNF	FOVF	0	FINXE	FINVE	FDBZE	FUNFE	FOVFE	FRMC	

SPR - 512; Read/Write; Reset - 0x0

Figure 15-6. LSP/EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in the following table.

Table 15-17. LSP/EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z4251n3 Supports only mode 0.

Table continues on the next page...

Table 15-17. LSP/EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
		Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by e200 core)
16 (48)	SOV	Summary integer overflow Defined by LSP APU.
17 (49)	OV	Integer overflow Defined by LSP APU.
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler. FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG l FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled

Table continues on the next page...

Table 15-17. LSP/EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
		1 Exception enabled
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 (62:63)	FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

15.7 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

15.7.1 Cache overview

The e200z4251n3 processor supports an 8 KB 2-way set-associative instruction and 4KB 2-way set-associative data cache with a 32-byte line size. The caches improves system performance by providing low-latency data to the e200z4251n3 instruction and data pipelines, which decouples processor performance from system memory performance.

The e200z4251n3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction and data addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

15.7.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache and for disabling ways in both caches. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR0 register is shown in the following figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
WID		0		WDD		0						DCWA	0		DCECE	DCEI	DCLOC		0						DCEA		DCLOINV	0	DCABT	DCINV	DCE

SPR - 1010; Read/Write; Reset - 0x0

Figure 15-7. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 fields are described in the following table.

Table 15-18. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:3	—	Reserved
4:5	WDD	Way Data Disable Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache. 0 The corresponding way in the data cache is available for replacement by data miss line fills.

Table continues on the next page...

Table 15-18. L1CSR0 field descriptions (continued)

Bits	Name	Description
		1 The corresponding way in the data cache is not available for replacement by data miss line fills.
8:11	—	Reserved[1]
12	DCWA	Data Cache Write Allocation Policy This bit also controls merging of store data into the linefill buffer while a cache linefill is in progress. Store data will not be merged when write allocation is disabled. 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled
13:14	—	Reserved[1]
15	DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	DCEI	Data Cache Error Injection DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits (p_dchk[0:1]). This includes writes due to store hits as well as writes due to cache line refills.
17:18	DCLOC	Data Cache Lockout Control **Note:** For the **dcbi** and **dcbf** instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. **Note:** When operating in MC mode (DCEA = 00), detected errors on cache-inhibited accesses will not cause a data cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	DCEA	Data Cache Error Action

Table continues on the next page...

Table 15-18. L1CSR0 field descriptions (continued)

Bits	Name	Description
		00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	DCLOINV	Data Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing DCINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	DCINV	Data Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. 0 No cache invalidate 1 Cache invalidation operation
31	DCE	Data Cache Enable When disabled, cache lookups are not performed for normal load or store accesses.Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE. 0 Cache is disabled 1 Cache is enabled

[1] These bits are not implemented and should be written with zero for future compatibility.

15.7.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR1 register is shown in the following figure.

0	ICECE	ICEI	ICLOC	0	ICEA	ICLOINV	0	ICABT	ICINV	ICE
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14	15	16	17 18	19 20 21 22 23 24	25 26	27	28	29	30	31

SPR - 1011; Read/Write; Reset - 0x0

Figure 15-8. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 fields are described in the following table.

Table 15-19. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits.
17:18	ICLOC	Instruction Cache Lockout Control **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. See ECC/EDC Error Handling for Cache Control Operations and Instructions. **Note:** When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out.

Table continues on the next page...

Table 15-19. L1CSR1 field descriptions (continued)

Bits	Name	Description
		10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate

Table continues on the next page...

Table 15-19. L1CSR1 field descriptions (continued)

Bits	Name	Description
		1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

15.7.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z4251n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The L1CFG0 register is shown in the following figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CARCH		CWPA	CFAHA	DCFISWA	0		DCBSIZE		DCREPL		DCLA	DCECA	DCNWAY								DCSIZE										
0	0	1	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0

SPR - 515; Read-only

Figure 15-9. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 bits are described in the following table.

Table 15-20. L1CFG0 field descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available 1 The caches support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 The data cache supports invalidation by Set and Way via L1FINV0
5:6	—	Reserved - read as zeros

Table continues on the next page...

Table 15-20. L1CFG0 field descriptions (continued)

Bits	Name	Description
7:8	DCBSIZE	Data Cache Block Size 00 The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11 The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x004 The size of the data cache is 4 KB

15.7.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z4251n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The L1CFG1 register is shown in the following figure.

<table>
<tr><td colspan="4">0</td><td>ICFISWA</td><td colspan="2">0</td><td colspan="2">ICBSIZE</td><td colspan="2">ICREPL</td><td>ICLA</td><td>ICECA</td><td colspan="8">ICNWAY</td><td colspan="11">ICSIZE</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
</table>

SPR - 516; Read-only

Figure 15-10. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 fields are described in the following table.

Table 15-21. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size

Table continues on the next page...

Table 15-21. L1CFG1 field descriptions (continued)

Bits	Name	Description
		00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 The size of the instruction cache is 8 KB

15.7.6 Data Cache Software Coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

15.7.7 Data Cache Hardware Coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency. Debug support is provided, however, for a D-Cache line invalidation operation requested by an external hardware debugger. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available D-Cache access cycle.

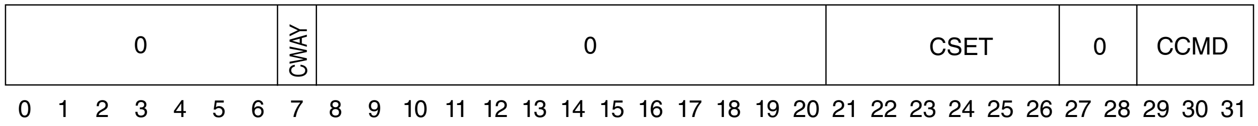
15.7.8 Cache Invalidate by Set and Way

e200z4251n3 supports cache set/way invalidation under software control. The caches may be invalidated by index and way through a **mtspr l1finv{0,1**} instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0,1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

15.7.8.1 L1FINV0

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in the following figure.



SPR - 1016; Read/Write; Reset - 0x0

Figure 15-11. L1 Flush/Invalidate Register 0 (L1FINV0)

The L1FINV0 bits are described in the following table.

Table 15-22. L1FINV0 field descriptions

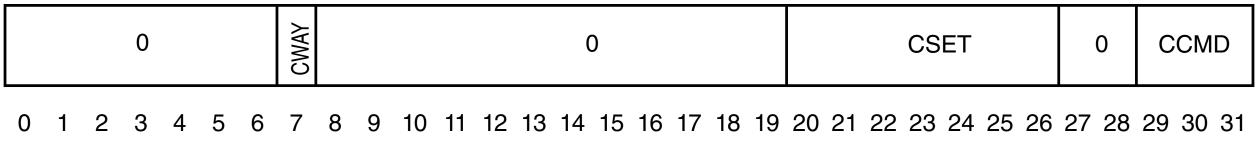
Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way Specifies the data cache way to be selected
8: 20	—	Reserved[1] for set extension
21 :26	CSET	Cache Set Specifies the cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 = The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

[1] These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

15.7.8.2 L1FINV1

The L1FINV1 register is shown in following figure.



SPR - 959; Read/Write; Reset - 0x0

Figure 15-12. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in the following table.

Table 15-23. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8: 19	—	Reserved[1] for set extension
20 :26	CSET	Cache Set Specifies the instruction cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

[1] These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

15.7.9 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of each cache. Seven parity check bits are provided for each tag entry for the tag arrays of both caches to support error detection and correction (ECC - SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, and each word in the data arrays of the D-Cache, which are used for single- and double-bit error detection (EDC - DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the L1CSR0[DCECE] and L1CSR1[ICECE] control fields. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the respective cache when cache error checking is disabled for that cache ($L1CSR\{0,1\}_{[D,I]CECE} = 0$).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, normal loads, or normal stores a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Store hits to the D-Cache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full-width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the D-Cache during otherwise idle cycles when two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

15.7.9.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the I-Cache and D-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by L1CSR{0,1}$_{\text{[D,I]CLOC}}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on normal instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via L1CSR{0,1} $_{\text{[D,I]CLOC}}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch or normal load or store access, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, to avoid certain exception conditions and for consistency with error reporting, specialized load/store accesses do not set LO bits; instead, cache contents are ignored for lines with those errors. Cache flush and invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable normal access will perform the lockout function if required.

15.7.10 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If L1CSR1$_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If L1CSR0$_{DCEI}$ is set, any cache line fill to the data cache data array has the associated two most significant parity check bits inverted in the data array for each word loaded. Additionally, inverted parity bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction or an L1FINV{0,1} invalidation operation.

15.8 Exceptions

Interrupts implemented in e200z4251n3 and the exception conditions that cause them are listed in the following table.

Table 15-24. Exceptions and Conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00	**p_critint_b** is asserted and MSR$_{CE}$ = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. Parity Error signaled on Local Memory access 5. External bus error 6. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40[1]	Interrupt Controller interrupt and MSR$_{EE}$ = 1

Table continues on the next page...

Table 15-24. Exceptions and Conditions (continued)

Interrupt type	Interrupt vector offset value	Causing conditions
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3. LSP ld and st instructions not properly aligned 4. **dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/PMGC0$_{UDI}$ = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/PMGC0$_{UDI}$ = 1
EFPU Data Exception	0xA0	Embedded Floating-point Data Exception
EFPU Round Exception	0xB0	Embedded Floating-point Round Exception
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

15.8.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				PIL	PPR	PTR	FP	ST	0			0	0	BO	0	0								SPV	0	VLEMI	0			0	0

SPR - 62; Read/Write; Reset - 0x0

Figure 15-13. Exception Syndrome Register (ESR)

The ESR bits are defined in the following table.

Table 15-25. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For e200z4251n3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program

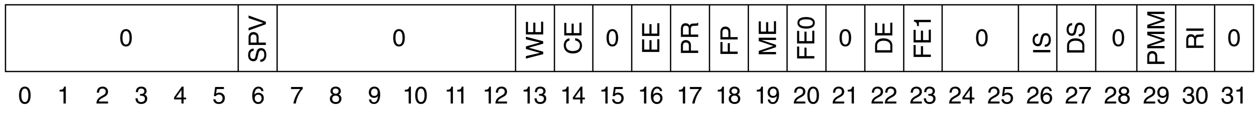
Table continues on the next page...

Table 15-25. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

15.8.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z4251n3 MSR is shown in the following figure.



Read/Write; Reset - 0x0

Figure 15-14. Machine State Register (MSR)

The MSR bits are defined in the following table.

Table 15-26. MSR field descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	SPV	SP/Embedded FP/Vector available 0 SP/Embedded FP Double/Vector unit is unavailable. The processor cannot execute SP/ Embedded FP Double or Vector Unit instructions. 1 SP/Embedded FP Double/Vector unit is available. The processor can execute SP/ Embedded FP Double or Vector Unit instructions. **NOTE:** The e200z4251n3 processor does not support these units in hardware. An Illegal Instruction exception is generated for attempted execution of these instructions regardless of the setting of SPV. SPV is ignored, but cleared on exceptions.
7:12	—	Reserved
13	WE	Wait State (Power management) enable. **NOTE:** this bit is implementation dependent and is being phased out. It will be removed in a future release. Currently it is implemented as a writeable bit, and is ignored, but cleared on exceptions.
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	FP	Floating-Point Available 0 Floating point unit is unavailable. The processor cannot execute floating-point instructions, including floating-point loads, stores, and moves. 1 Floating Point unit is available. The processor can execute floating-point instructions. **NOTE:** The e200z4251n3 processor does not support the PowerISA 2.06 floating point unit in hardware, and an Illegal Instruction exception is generated for attempted execution of PowerISA 2.06 floating point instructions regardless of the setting of FP. FP is ignored, but cleared on exceptions.
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 (not used) **NOTE:** The e200z4251n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE0 is ignored, but cleared on exceptions.

Table continues on the next page...

Table 15-26. MSR field descriptions (continued)

Bit(s)	Name	Description
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 (not used) **NOTE:** The e200z4251n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE1 is ignored, but cleared on exceptions.
24:25	—	Reserved
26	IS	Instruction Address Space 0 The processor directs all instruction fetches to address space 0. 1 The processor directs all instruction fetches to address space 1. **NOTE:** The e200z4251n3 processor does not support Address Spaces, thus the IS bit is ignored, but cleared on exceptions.
27	DS	Data Address Space 0 The processor directs all data storage accesses to address space 0. 1 The processor directs all data storage accesses to address space 1. **NOTE:** The e200z4251n3 processor does not support Address Spaces, thus the DS bit is ignored, but cleared on exceptions.
28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

15.8.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in the following figure.

MCP	IC_DPERR	0	DC_DPERR	EXCP_ERR	IC_TPERR	DC_TPERR	IC_LKERR	DC_LKERR	0		NMI	MAV	MEA	U	IF	LD	ST	G	0		STACK_ERR	IMEM_PERR	DMEM_RDPERR	DMEM_WRPERR	0		BUS_IRERR	BUS_DRERR	BUS_WRERR	BUS_WRDSI	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 572; Read/Clear; Reset - 0x0

Figure 15-15. Machine Check Syndrome Register (MCSR)

The following table describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as "write 1 to clear." Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling $\mathrm{MSR}_{\mathrm{ME}}$ to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once $\mathrm{MSR}_{\mathrm{ME}} = 1$.

Table 15-27. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type[1]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid	Status	—

Table continues on the next page...

Table 15-27. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
		Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition		
13	MEA	MCAR holds Effective Address If MAV = 1, MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address[2]	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR[3]	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error	Async Mchk	Maybe

Table continues on the next page...

Table 15-27. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
		Indicates an uncorrectable error in the DMEM on a CPU port write access.		
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.
   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "Status" indicates that this bit provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a '1' to the bit position.
   - "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME} = 1. If MSR_{ME} = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
2. Note that since no address translation mechanism is present on e200z4251n3, the MEA value is always set to '0'.
3. Will not be set by e200z4251n3 since no local instruction memory is present.

15.8.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in the following figure.

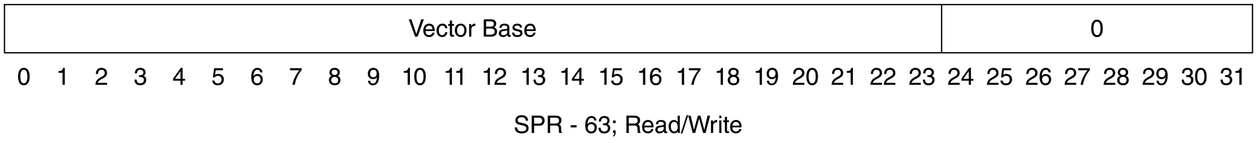


Figure 15-16. Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in the following table.

Table 15-28. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256-byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

15.8.5 Interrupt Definitions

15.8.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

The following table lists register settings when a Critical Input interrupt is taken.

Table 15-29. Critical Input Interrupt—register settings

<table>
<tr><th>Register</th><th colspan="6">Setting description</th></tr>
<tr><td>CSRR0</td><td colspan="6">Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.</td></tr>
<tr><td>CSRR1</td><td colspan="6">Set to the contents of the MSR at the time of the interrupt</td></tr>
<tr><td rowspan="5">MSR</td><td>SPV</td><td>0</td><td>FP</td><td>0</td><td>FE1</td><td>0</td></tr>
<tr><td>WE</td><td>0</td><td>ME</td><td>—</td><td>IS</td><td>0</td></tr>
<tr><td>CE</td><td>0</td><td>FE0</td><td>0</td><td>DS</td><td>0</td></tr>
<tr><td>EE</td><td>0</td><td>DE</td><td>—/0[1]</td><td>PMM</td><td>0</td></tr>
<tr><td>PR</td><td>0</td><td></td><td></td><td>RI</td><td>—</td></tr>
<tr><td>ESR</td><td colspan="6">Unchanged</td></tr>
<tr><td>MCSR</td><td colspan="6">Unchanged</td></tr>
<tr><td>DEAR</td><td colspan="6">Unchanged</td></tr>
<tr><td>Vector</td><td colspan="6">$IVPR_{0:23}$ || 0x00 (autovectored)
$IVPR_{0:15}$ || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)</td></tr>
</table>

[1] Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

15.8.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in the following table.

Table 15-30. Machine Check Interrupt—register settings

Register	Setting description					
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.					
MCSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—[1]	PMM	0
	PR	0			RI	0
ESR	Unchanged					
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.					
Vector	$IVPR_{0:23}$ || 0x10					

[1] Clearing of DE is optionally supported by control in HID0.

15.8.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

The following table lists register settings when a DSI is taken.

Table 15-31. Data Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.					
Vector	IVPR$_{0:23}$ || 0x20					

15.8.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

The following table lists register settings when an ISI is taken.

Table 15-32. Instruction Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	IVPR$_{0:23}$ || 0x30					

15.8.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

The following table lists register settings when an External Input interrupt is taken.

Table 15-33. External Input Interrupt—Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

15.8.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.
- Execution of an LSP APU load or store instruction that is not properly aligned.

The following table lists register settings when an alignment interrupt is taken.

Table 15-34. Alignment Interrupt—register settings

<table>
<tr><th>Register</th><th colspan="6">Setting description</th></tr>
<tr><td>SRR0</td><td colspan="6">Set to the effective address of the excepting load/store/dcbz instruction.</td></tr>
<tr><td>SRR1</td><td colspan="6">Set to the contents of the MSR at the time of the interrupt</td></tr>
<tr><td rowspan="5">MSR</td><td>SPV</td><td>0</td><td>FP</td><td>0</td><td>FE1</td><td>0</td></tr>
<tr><td>WE</td><td>0</td><td>ME</td><td>—</td><td>IS</td><td>0</td></tr>
<tr><td>CE</td><td>—</td><td>FE0</td><td>0</td><td>DS</td><td>0</td></tr>
<tr><td>EE</td><td>0</td><td>DE</td><td>—</td><td>PMM</td><td>0</td></tr>
<tr><td>PR</td><td>0</td><td></td><td></td><td>RI</td><td>—</td></tr>
<tr><td>ESR</td><td colspan="6">[ST], [SPV], VLEMI. All other bits cleared.</td></tr>
<tr><td>MCSR</td><td colspan="6">Unchanged</td></tr>
<tr><td>DEAR</td><td colspan="6">Set to the effective address of a byte of the load or store access causing the violation.</td></tr>
<tr><td>Vector</td><td colspan="6">IVPR$_{0:23}$ || 0x50</td></tr>
</table>

15.8.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z4251n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z4251n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when MSR$_{PR}$ = 1 (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with SPRN$_5$ = 1 (even if the SPR is undefined)

The e200z4251n3 will invoke a Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *Power ISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

The following table lists register settings when a Program interrupt is taken.

Table 15-35. Program Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented:		PIL, VLEMI. All other bits cleared.			
	Privileged:		PPR, VLEMI. All other bits cleared.			
	Trap:		PTR, VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x60					

15.8.5.8 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- $PMLCa x_{CE}$ = 11; that is, for the given counter the overflow condition is enabled.
- $PMC x_{OV}$ = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

The following table lists register settings when an performance monitor interrupt is taken.

Table 15-36. Performance Monitor Interrupt—Register Settings

Register	Setting description					
SRR0/ DSRR0	Set to the effective address of the next instruction to be executed.					
SRR1/ DSRR1[1]	Set to the contents of the MSR at the time of the interrupt.					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	0/—[3]	DE	—/0[4]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	IVPR$_{0:23}$ || 0x70					

1. DSRR0/1 are used if PMGC0$_{UDI}$ = 1
2. CE is cleared if PMGC0$_{UDI}$ = 1 and HID0$_{DCLRCE}$ = 1
3. EE is not cleared if PMGC0$_{UDI}$ = 1 and HID0$_{DCLREE}$ = 0
4. DE is cleared if PMGC0$_{UDI}$ = 1

15.8.5.9 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

The following table lists register settings when a System Call interrupt is taken.

Table 15-37. System Call Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—

Table continues on the next page...

Table 15-37. System Call Interrupt—register settings (continued)

Register	Setting description
ESR	VLEMI. All other bits cleared.
MCSR	Unchanged
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x80

15.8.5.10 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM}$ = 1 (internal debug mode) and MSR_{DE} = 1).

The following table lists register settings when a Debug interrupt is taken.

Table 15-38. Debug Interrupt—register settings

Register	Setting description					
DSRR0[1]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
DSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0	FE0	0	DS	0
	EE	—/0[2]	DE	0	PMM	0
	PR	0			RI	—
DBSR[3]	Unconditional Debug Event:	UDE				
	Instr. Complete Debug Event:	ICMP				
	Branch Taken Debug Event:	BRT				
	Interrupt Taken Debug Event:	IRPT				
	Critical Interrupt Taken Debug Event:	CIRPT				
	Trap Instruction Debug Event:	TRAP				
	Instruction Address Compare:	{IAC}				
	Data Address Compare:	{DACR, DACW}				
	Debug Notify Interrupt:	DNI				
	Return Debug Event:	RET				
	Critical Return Debug Event:	CRET				

Table continues on the next page...

Table 15-38. Debug Interrupt—register settings (continued)

Register	Setting description	

	External Debug Event:
	Performance Monitor Debug Event:
	MPU Debug Event:
	and optionally, an
	Imprecise Debug Event flag
		{DEVT1, DEVT2}
		PMI
		MPU
		
		{IDE}

ESR	Unchanged	
MCSR	Unchanged	
DEAR	Unchanged	
Vector	$\text{IVPR}_{0:23}$ || 0x90	

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

15.8.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

The following table lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 15-39. Embedded Floating-point Data Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$\text{IVPR}_{0:23}$ || 0xA0					

15.8.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

The following table lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 15-40. Embedded Floating-point Round Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$\text{IVPR}_{0:23}$ || 0xB0					

15.8.5.13 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

The following table lists register settings when a System Reset interrupt is taken.

Table 15-41. System Reset Interrupt—register settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0

Table continues on the next page...

Table 15-41. System Reset Interrupt—register settings (continued)

Register	Setting description		
	CE 0	FE0 0	DS 0
	EE 0	DE 0	PMM 0
	PR 0		RI 0
ESR	Cleared		
DEAR	Undefined		
Vector	[**p_rstbase[0:29]**] ǁ 2'b00		

15.9 Memory Protection Unit (MPU)

15.9.1 MPU Overview

The e200z4251n3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

15.9.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

15.9.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure

mpu read entry

mpure

31	1	///	1 1 1 0 1 1 0 0 1 0	/
0 5	6	7 20	21 30	31

```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

15.9.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe

mpu write entry

mpuwe

31	1	///	1 1 1 1 0 1 0 0 1 0	/
0 5	6	7 20	21 30	31

```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

15.9.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the core.

mpusync　　mpusync

MPU Synchronize
mpusync

31	1	///	1 0 0 0 1 1 0 1 1 0	/
0　5	6	7　10　11　15　16　20	21　30	31

15.9.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z4251n3 MPU design. The MMUCFG register is shown in the following figure.

0	RASIZE	0	PIDSIZE	NMPUS	NTLBS	MAVN
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14	15 16 17 18 19 20	21 22 23 24 25	26 27	28 29	30 31

SPR - 1015; Read-Only

Figure 15-17. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in the following table.

Table 15-42. MMUCFG field descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU

Table continues on the next page...

Table 15-42. MMUCFG field descriptions (continued)

Bits	Name	Function
26:27	NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

15.9.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z4251n3 MPU. The MPU0CFG register is shown in following figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FASSOC	0							MINSIZE				MAXSIZE				IPROT	0	UAMSKA	0				SHENTRY			DENTRY			IENTRY		

SPR - 692; Read-Only

Figure 15-18. MPU0 Configuration Register (MPU0CFG)

The MPU0CFG bits are described in the following table.

Table 15-43. MPU0CFG field descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15	MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
17	—	Reserved

Table continues on the next page...

Table 15-43. MPU0CFG field descriptions (continued)

Bits	Name	Function
18	UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28	DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

15.9.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in the following figure.

0																BYPSR	BYPSW	BYPSX	BYPUR	BYPUW	BYPUX	0		DRDEN	DWDEN	IDEN	0	TIDCTL	0	MPUFI	MPUEN
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 1014; Read/ Write; Reset - 0x0[1]

Figure 15-19. MPU0 Control and Status Register 0 (MPU0CSR0)

Note

[1] Reset by processor reset **p_reset_b** if EDBCR0$_{EDM}$ = 0, as well as unconditionally by an internal power-on reset signal. If EDBCR0$_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Specifically, this applies to the DRDEN, DWDEN, and IDEN control bits.

The MPU0CSR0 bits are described in the following table.

Table 15-44. MPU0CSR0 field descriptions

Bits	Name	Description
0:15	—	Reserved

Table continues on the next page...

Table 15-44. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
16	BYPSR	Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17	BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19	BYPUR	Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses
20	BYPUW	Bypass MPU protections for User Write accessesThis bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21	BYPUX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
22:23	—	Reserved

Table continues on the next page...

Table 15-44. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
24	DRDEN	Data Read Debug Enable This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses
25	DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses 1 MPU debug events are enabled for data write accesses
26	IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses 1 MPU debug events are enabled for instruction accesses
27	—	Reserved
28	TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode
29	—	Reserved
30	MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. 0 No flash invalidate 1 MPU1 invalidation operation **Note:** Entries that have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable

Table 15-44. MPU0CSR0 field descriptions

Bits	Name	Description
		This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

15.9.9 MPU Assist Registers (MAS)

The core uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in Figure 15-20. Fields are defined in Table 15-45.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
VALID	IPROT	SEL		0	RO	DEBUG	INST	SHD	0			ESEL				0	UAMSK			UW	SW	UX/UR	SX/SR	IOVR	GOVR[1]	1	1	I	0	G[1]	0

SPR - 624; Read/ Write; Reset - Unaffected

Figure 15-20. MPU Assist Register 0 (MAS0)

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Table 15-45. MAS0—MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 = This MPU entry is invalid 1 = This MPU entry is valid
1	IPROT	Invalidation Protect 0 = Entry is not protected from invalidation 1 = Entry is protected from invalidation
2:3	SEL	Selects MPU for access: 00 = Reserved, no access 01 = Reserved, no access

Table continues on the next page...

Table 15-45. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
		10 = Select MPU 11 = Reserved, no access
4	—	Reserved
5	RO	Read-Only When set to 1, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs. 0 = This MPU entry is writable 1 = This MPU entry is Read-only
6	DEBUG	Debug Control for Entry This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access. 0 = This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 = This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access
7	INST	Instruction Entry When SHD = 0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations. When SHD = 1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. 0 = This MPU entry is used for matching data accesses only 1 = This MPU entry is used for matching instruction accesses only
8	SHD	Shared Entry Select This bit is used to control selection of the Shared portion of the region descriptor table. 0 = The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 = The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected.
9:11	—	Reserved
12 :15	ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved
17:19	UAMSK	Upper Address Mask Control

Table continues on the next page...

Table 15-45. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
		This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly. 000 = No upper address bits are masked, address matching uses all 32 address bits for accesses 001 = The most significant access address bit is forced to zero before matching 010 = The two most significant access address bits are forced to zero before matching 011 = The three most significant access address bits are forced to 0 before matching 100 = The four most significant access address bits are forced to 0 before matching 101 = The upper five access address bits are forced to zero before matching 11x = Reserved, do not use
20	UW	User Mode Write PermissionDetermines User mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No User mode Write permission 1 = User mode has Write permission
21	SW	Supervisor Mode Write / Read Permission Determines Supervisor mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No Supervisor mode Write permission 1 = Supervisor mode has Write permission
22	UX/UR	User Mode Execute / Read Permission Determines User mode Execute (X) permission when INST = 1, or User mode Read (R) permission when INST = 0. 0 = No User mode Execute/ Read permission 1 = User mode has Execute/ Read permission
23	SX/SR	Supervisor Mode Execute / Read Permission Determines Supervisor mode Execute (X) permission when INST = 1, or Supervisor mode Read (R) permission when INST = 0. 0 = No Supervisor mode Execute/ Read permission 1 = Supervisor mode has Execute/ Read permission
24	IOVR	Cache-Inhibit attribute OverrideDetermines whether this matching entry overrides the I bit settings of other matching entries and the cache-inhibited region configuration signals 0 = No override of I attribute by entry 1 = Entry I bit overrides I attribute
25	GOVR/—	G attribute Override Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = No override of G attribute by entry 1 = Entry G bit overrides G attribute

Table continues on the next page...

Table 15-45. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
26	—	Reserved
27	—	Reserved
28	I	Cache Inhibited This attribute may be overridden by the cache-inhibited region configuration input settings. When the core memory protection unit (CMPU) is disabled, the CMPU configuration is ignored and this bit is ignored also. 0 = This region is considered cacheable 1 = This region is considered cache-inhibited
29	—	Reserved
30	G/—	Guarded This attribute may be overridden by the guarded region configuration input settings.This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 = All loads and stores to this region are performed without speculation (i.e. they are known to be required)
31	—	Reserved

The MAS1 register is shown in Figure 15-21. Fields are defined in Table 15-46.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0								TID								0								TIDMSK							

SPR - 625; Read/Write; Reset - Unaffected

Figure 15-21. MPU Assist Register 1 (MAS1)

Table 15-46. MAS1—Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx = TID[0] comparison to PID0[0] not masked 1xxxxxxx = TID[0] comparison to PID0[0] is masked x0xxxxxx = TID[1] comparison to PID0[1] not masked x1xxxxxx = TID[1] comparison to PID0[1] is masked

Table 15-46. MAS1—Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set

		xx0xxxxx = TID[2] comparison to PID0[2] not masked
		xx1xxxxx = TID[2] comparison to PID0[2] is masked
		xxx0xxxx = TID[3] comparison to PID0[3] not masked
		xxx1xxxx = TID[3] comparison to PID0[3] is masked
		xxxx0xxx = TID[4] comparison to PID0[4] not masked
		xxxx1xxx = TID[4] comparison to PID0[4] is masked
		xxxxx0xx = TID[5] comparison to PID0[5] not masked
		xxxxx1xx = TID[5] comparison to PID0[5] is masked
		xxxxxx0x = TID[6] comparison to PID0[6] not masked
		xxxxxx1x = TID[6] comparison to PID0[6] is masked
		xxxxxxx0 = TID[7] comparison to PID0[7] not masked
		xxxxxxx1 = TID[7] comparison to PID0[7] is masked
		This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in Figure 15-22. Fields are defined in Table 15-47.

UPPER BOUND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 626; Read/Write; Reset - Unaffected

Figure 15-22. MPU Assist Register 2 (MAS2)

Table 15-47. MAS2—Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in Figure 15-23. Fields are defined in Table 15-48.

LOWER BOUND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 627; Read/Write; Reset - Unaffected

Figure 15-23. MPU Assist Register 3 (MAS3)

Table 15-48. MAS3—Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

15.9.10 MAS Registers Summary

The MAS registers are summarized in the following figure.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MAS0	VALID	IPROT	SEL		0	RO	DEBUG	INST	SHD	0			ESEL				0	UAMSK			UW	SW	UX/UR	SX/SR	IOVR	GOVR[1]	1	1	I	0	G[1]	0
MAS1	0								TID								0								TIDMSK							
MAS2	UPPER BOUND																															
MAS3	LOWER BOUND																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Figure 15-24. MPU Assist Registers Summary

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

15.10 Local memories

15.10.1 Local Data memory overview

Local Data (DMEM) memory has been added to provide low latency access for critical instruction routines and data operands. Access latency (zero wait-states) is similar to that of a data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large capacity tightly coupled storage.

The local memory has one dedicated port for CPU accesses and another shared slave port for external accesses. DMEM provides a dedicated port to the CPU's load/store unit. Access via the dedicated CPU port is referred to as *front door* access. Access via the slave port is referred to as *back door* access.

Local memory occupies a portion of the processor's address space and is conditionally accessed in parallel with the corresponding cache (if present) when a CPU request is made, based on an address decode comparison early in the access pipeline.

External masters use the shared slave port, which is implemented as an AHB 2.v6 slave interface, to access the DMEM. Slave port accesses are arbitrated with the CPU port in a round-robin fashion to prevent starvation from occurring on either port.

NOTE

DMEM, by default, is always accessed through the front door (CPU) interface, regardless of whether the program uses Decorated Access, Non-decorated Cache Bypass, Atomic Load and Reserve instructions, as well as regular load/store instructions (stw, lwz, for example). The only exception is when the core's DMEMCTL0 register is set to bypass the type of DMEM access used onto AHB back door slave interface. Regular load/store instructions always access DMEM through the front door interface since no bypass control is available for these kinds of instructions.

15.10.2 Local memory control and configuration

DMEM local memory configuration information is provided by a set of privileged read-only SPRs that are accessed using **mfspr** instructions. DMEM local memory operation is controlled by a set of privileged device control registers (DCRs) that are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

15.10.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z4251n3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in the following figure.

DMEM BASE ADDR	0	DECUA	DECA	0	DMSIZE	0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	20	21	22	23	24 25 26 27 28 29	30 31
CURRENT DMEM BASE ADDRESS VALUE	0	1	1	0	010000 (64 KB)	00

SPR - 694; Read-only; Privileged

Figure 15-25. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in the following table.

Table 15-49. DMEMCFG0 field descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0s when the DMEM size is > 4 KB
20	—	Reserved
21	DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available. 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access.
22	DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available. 1 Error Correction is available.
23	—	Reserved
24:29	DMSIZE	DMEM Size 010000 The size of the DMEM is 64 KB.
30:31	—	Reserved

15.10.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DMEM BASE ADDR																0			DBYPCB	DBYPAT	DBYPDEC	DECUE	0	DBAPD	DPEIE	DICWE	DSWCE		DDAUEC	DCPECE	DSECE

DCR - 496; Read/Write; Reset[1] - 19‘b0 || p_dmem_rstcfg[8,6:7,0] || 4‘b0 || p_dmem_rstcfg[1:5]; Privileged

Figure 15-26. DMEM Control Register 0 (DMEMCTL0)

Note

[1] Reset by an internal power-on reset signal or by an internal destructive reset signal. Unaffected by **p_reset_b**.

Table 15-50. DMEMCTL0 field descriptions

Bits	Name	Description
0:15	DMEM BASEADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that no cached entries are present in the D-Cache for the memory space to be occupied by the DMEM.
16 :18	—	Reserved
19	DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses that target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port. 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[8]** input, which is internally hard coded.
20	DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems that support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Atomic CPU accesses do not bypass the DMEM CPU port. 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface.

Table continues on the next page...

Table 15-50. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		Note: This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[6]** input, which is internally hard coded.
21	DBYPDEC	DMEM Bypass Decorated CPU accesses DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems that support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port. 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[7]** input, which is internally hard coded.
22	DECUE	DMEM Error Correction Update Enable DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM. 0 Error Correction Update is disabled. 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be rewritten with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.) **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[0]** input which is internally hard coded.
23	—	Reserved
24	DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25	DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled. 1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits.
26	DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set.

Table continues on the next page...

Table 15-50. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
		0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).
27:28	DSWCE	DMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the values of the **p_dmem_rstcfg[1:2]** inputs, which are internally hard coded.
29	DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[3]** input, which is an internally hardcoded signal.
30	DCPECE	DMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[4]** input, which is internally hard coded.
31	DSECE	DMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written.

Table 15-50. DMEMCTL0 field descriptions

Bits	Name	Description
		1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. **Note:** This field is updated by Reset from an internal power-on reset signal or by an internal destructive reset signal based on the value of the **p_dmem_rstcfg[5]** input, which is an internally hard coded signal.

15.10.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed. All other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses that target DMEM address range, regardless of whether access may bypass the DMEM based on settings in $\text{DMEMCTL0}_{\text{DBYPCB, DBYPATF, DBYPDEC}}$.

0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31
DSWAC3	DSWAC2	DSWAC1	DSWAC0	DSRAC3	DSRAC2	DSRAC1	DSRAC0	DUWAC3	DUWAC2	DUWAC1	DUWAC0	DURAC3	DURAC2	DURAC1	DURAC0

DCR - 498; Read/Write; Reset[1] - 32‘b0; Privileged

Figure 15-27. DMEM Control Register 1 (DMEMCTL1)

Note

[1] Reset from an internal power-on reset signal or by an internal reset signal.

Table 15-51. DMEMCTL1 field descriptions

Bits	Name	Description
0:1	DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 15-51. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11
2:3	DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10
4:5	DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01
6:7	DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2'b00
8:9	DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2'b11

Table continues on the next page...

Table 15-51. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
10:11	DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2'b10
12:13	DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2'b01
14:15	DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2'b00
16:17	DUWAC3	DMEM User Write Access Control for Quadrant 3 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2'b11.
18:19	DUWAC2	DMEM User Write Access Control for Quadrant 2 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM

Table continues on the next page...

Table 15-51. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2'b10
20:21	DUWAC1	DMEM User Write Access Control for Quadrant 1 00 = User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 = User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 = User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 = Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2'b01
22:23	DUWAC0	DMEM User Write Access Control for Quadrant 0 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00
24:25	DURAC3	DMEM User Read Access Control for Quadrant 3 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11.
26:27	DURAC2	DMEM User Read Access Control for Quadrant 2 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved

Table continues on the next page...

Table 15-51. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
		Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10
28:29	DURAC1	DMEM User Read Access Control for Quadrant 1 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01
30:31	DURAC0	DMEM User Read Access Control for Quadrant 0 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00

15.11 End-to-End ECC Support

This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

15.11.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

15.11.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0																												IDAC	IECCEN	DDAC	DECCEN

DCR - 510; Read/Write; Reset - 0x5 || p_e2e_rstcfg[0:3]; Privileged

Figure 15-28. e2eECC Control Register 0 (E2ECTL0)

Table 15-52. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices.

Table 15-52. E2ECTL0 field descriptions

Bits	Name	Description
		1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $\text{E2ECTL0}_{\text{[I,D]ECCEN}}$ control bits.

The $\text{E2ECTL0}_{\text{[I,D]DAC}}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

15.11.1.2 End-to-End ECC fault injection by software

Fault injection provides a way to test error recovery and portions of the error detection/ correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus (but not on cache hits). For the case of an external read access that receives an ERROR response (HRESP=ERR), the RCHKBIT field is written with all zeros.

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

The following figure shows the layout of E2EECSR0.

0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
0 1 2 3	4 5 6 7 8 9 10 11	12 13 14 15	16 17	18 19	20 21 22 23	24 25 26 27 28 29 30 31

DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 15-29. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 15-53. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked). If the CPU receives an ERROR response (HRESP=ERR) on an external read access, the RCHKBIT field is written with all zeros.
12:15	—	Reserved
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved Note: Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation.

Table continues on the next page...

Table 15-53. E2EECSR0 field descriptions (continued)

Bits	Name	Description
		10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further fault injection. 11 Reserved Note: Checkbit substitution has priority over Error injection
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

Chapter 16
System Integration Unit Lite2 (SIUL2)

16.1 Introduction

16.1.1 Overview

The System Integration Unit Lite2 provides control over all the electrical pin controls and as many as 32 ports with 16 bits of bidirectional, general-purpose input and output signals. One of the most important functions of the SIUL2 is that it enables the user to select the functions and electrical characteristics that appear on external device pins. It also controls the multiplexing of internal signals from one module to another and controls device I/O. It supports as many as 32 external interrupts with trigger event configuration. The following figure is the block diagram of SIUL2 and its interfaces to other system components.

This module provides dedicated pad control to general-purpose pads that can be configured as either inputs or outputs. The SIUL2 module provides registers that enable user software to read values from GPIO pads configured as inputs, and write values to GPIO pads configured as outputs:

- When configured as output, you can write to an internal register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from an internal register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the written value appeared on the pad.

To assist software development, GPIO data registers can be accessed using various mechanisms. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation.

- Access to two 16-bit ports in one access
- Read/write access to a single bit
- A 16-bit port write with a bit mask, using single 32-bit access.

The external interrupt sources can be configured at the MCU level to be used with any chip pad. From 1 to 32 of the interrupt sources can be configured to have a digital filter to reject short glitches on the inputs. The external interrupt/DMA requests map to the REQ pins in the device packages.

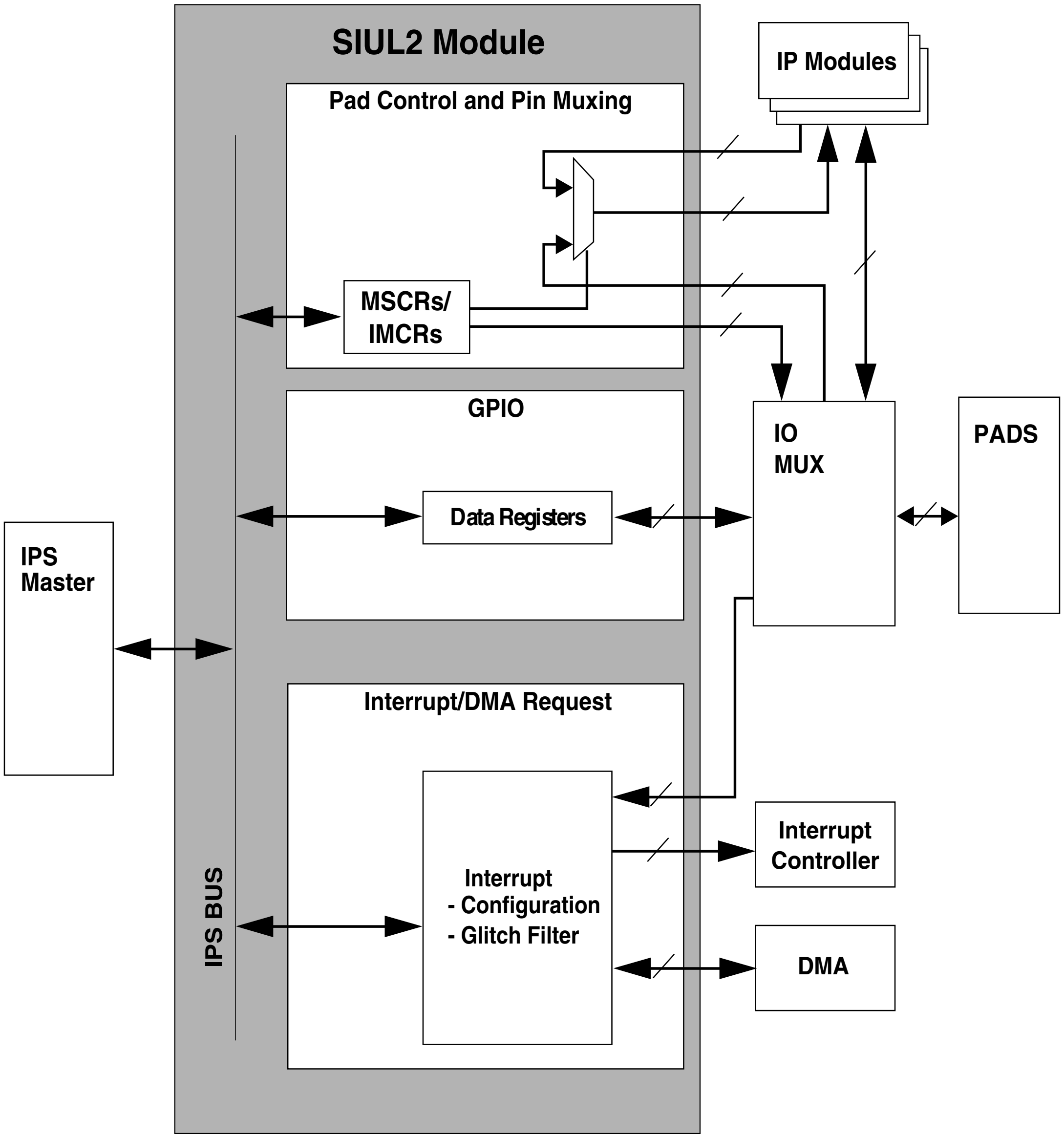


Figure 16-1. System Integration Unit Lite2 block diagram

16.1.2 Features

The System Integration Unit Lite2 supports these distinctive features:

- 1 to 32 GPIO ports with data control
    - Drive data to as many as 16 independent I/O channels
    - Sample data from as many as 16 independent I/O channels

Two 16-bit registers can be read/written with one access for a 32-bit port, if needed.

External interrupt/DMA request support with:

- 1 to 4 system interrupt vectors for 1 to 4 interrupt sources with independent interrupt masks. For 32 external interrupt sources (REQ pins), four groups have eight interrupt sources each, and each of the four groups is assigned one system interrupt vector.
- 1 to 32 programmable digital glitch filters, one for each REQ pin
- 1 to 4 system DMA request channels for 1 to 4 REQ pins
- Edge detection

Additionally the SIUL2 contains the Multiplexed Signal Configuration Registers (MSCR) that configure the electrical parameters and settings for as many as 512 functional pads. The number of these registers that is actually implemented varies by device. These registers configure the following pad features:

- Drive strength
- Output impedance control
- Open drain/source output enable
- Slew rate control
- Hysteresis control
- Internal pull control and pull selection
- Pin function assignment
- Control of analog path switches
- Safe mode behavior configuration

16.1.3 Register protection

The System Integration Unit Lite2 uses the Register Protection Scheme to protect the individual registers from accidental writes. The following registers can be protected:

- SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)
- SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)
- SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)
- SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)
- SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)
- SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0 to SIUL2_IFMCR31)
- SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)
- SIUL2 Multiplexed Single Configuration Register (SIUL2_MSCR0 to SIUL2_MSCR511 and SIUL2_IMCR0 to SIUL2_IMCR511): Only the specific MSCRs and IMCRs that are actually implemented on a device are protected.

A compiled list identifies the protected registers at specific addresses for this device.

16.2 Memory map and register description

This section provides a detailed description of all registers accessible in the SIUL2 module in address order. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

The following table lists the implemented SIUL2 registers.

NOTE

> Undocumented register spaces in the SIUL2 memory map, including addresses shown as "blank," are reserved. Reserved registers read as 0, and writes to these registers have no effect. If supported and enabled by the MCU, a transfer error is issued for attempts to access a completely reserved register space unless it is an 8 bit access. Do not expect transfer errors on 8 bit

accesses to reserved register space. A write to a reserved register bit has no effect.

SIUL2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	SIUL2 MCU ID Register #1 (SIUL2_MIDR1)	32	R	See section	16.2.1/471
8	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	32	R	See section	16.2.2/472
10	SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)	32	w1c	0000_0000h	16.2.3/473
18	SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)	32	R/W	0000_0000h	16.2.4/478
20	SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)	32	R/W	0000_0000h	16.2.5/482
28	SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)	32	R/W	0000_0000h	16.2.6/486
30	SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	32	R/W	0000_0000h	16.2.7/489
38	SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)	32	R/W	0000_0000h	16.2.8/492
40	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0)	32	R/W	0000_0000h	16.2.9/495
44	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR1)	32	R/W	0000_0000h	16.2.9/495
48	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR2)	32	R/W	0000_0000h	16.2.9/495
4C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR3)	32	R/W	0000_0000h	16.2.9/495
50	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR4)	32	R/W	0000_0000h	16.2.9/495
54	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR5)	32	R/W	0000_0000h	16.2.9/495
58	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR6)	32	R/W	0000_0000h	16.2.9/495
5C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR7)	32	R/W	0000_0000h	16.2.9/495
60	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR8)	32	R/W	0000_0000h	16.2.9/495
64	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR9)	32	R/W	0000_0000h	16.2.9/495
68	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR10)	32	R/W	0000_0000h	16.2.9/495
6C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR11)	32	R/W	0000_0000h	16.2.9/495
70	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR12)	32	R/W	0000_0000h	16.2.9/495
74	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR13)	32	R/W	0000_0000h	16.2.9/495

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
78	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR14)	32	R/W	0000_0000h	16.2.9/495
7C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR15)	32	R/W	0000_0000h	16.2.9/495
80	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR16)	32	R/W	0000_0000h	16.2.9/495
84	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR17)	32	R/W	0000_0000h	16.2.9/495
88	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR18)	32	R/W	0000_0000h	16.2.9/495
8C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR19)	32	R/W	0000_0000h	16.2.9/495
90	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR20)	32	R/W	0000_0000h	16.2.9/495
94	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR21)	32	R/W	0000_0000h	16.2.9/495
98	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR22)	32	R/W	0000_0000h	16.2.9/495
9C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR23)	32	R/W	0000_0000h	16.2.9/495
A0	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR24)	32	R/W	0000_0000h	16.2.9/495
A4	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR25)	32	R/W	0000_0000h	16.2.9/495
A8	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR26)	32	R/W	0000_0000h	16.2.9/495
AC	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR27)	32	R/W	0000_0000h	16.2.9/495
B0	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR28)	32	R/W	0000_0000h	16.2.9/495
B4	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR29)	32	R/W	0000_0000h	16.2.9/495
B8	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR30)	32	R/W	0000_0000h	16.2.9/495
BC	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR31)	32	R/W	0000_0000h	16.2.9/495
C0	SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)	32	R/W	0000_0000h	16.2.10/496
240	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0)	32	R/W	See section	16.2.11/497
244	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR1)	32	R/W	See section	16.2.11/497
248	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR2)	32	R/W	See section	16.2.11/497

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR3)	32	R/W	See section	16.2.11/497
250	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR4)	32	R/W	See section	16.2.11/497
254	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR5)	32	R/W	See section	16.2.11/497
258	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR6)	32	R/W	See section	16.2.11/497
25C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR7)	32	R/W	See section	16.2.11/497
260	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR8)	32	R/W	See section	16.2.11/497
264	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR9)	32	R/W	See section	16.2.11/497
268	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR10)	32	R/W	See section	16.2.11/497
26C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR11)	32	R/W	See section	16.2.11/497
270	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR12)	32	R/W	See section	16.2.11/497
274	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR13)	32	R/W	See section	16.2.11/497
278	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR14)	32	R/W	See section	16.2.11/497
27C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR15)	32	R/W	See section	16.2.11/497
280	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR16)	32	R/W	See section	16.2.11/497
284	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR17)	32	R/W	See section	16.2.11/497
288	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR18)	32	R/W	See section	16.2.11/497
28C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR19)	32	R/W	See section	16.2.11/497
290	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR20)	32	R/W	See section	16.2.11/497
294	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR21)	32	R/W	See section	16.2.11/497
298	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR22)	32	R/W	See section	16.2.11/497
29C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR23)	32	R/W	See section	16.2.11/497
2A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR24)	32	R/W	See section	16.2.11/497

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR25)	32	R/W	See section	16.2.11/497
2A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR26)	32	R/W	See section	16.2.11/497
2AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR27)	32	R/W	See section	16.2.11/497
2B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR28)	32	R/W	See section	16.2.11/497
2B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR29)	32	R/W	See section	16.2.11/497
2B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR30)	32	R/W	See section	16.2.11/497
2BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR31)	32	R/W	See section	16.2.11/497
2C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR32)	32	R/W	See section	16.2.11/497
2C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR33)	32	R/W	See section	16.2.11/497
2C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR34)	32	R/W	See section	16.2.11/497
2CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR35)	32	R/W	See section	16.2.11/497
2D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR36)	32	R/W	See section	16.2.11/497
2D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR37)	32	R/W	See section	16.2.11/497
2D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR38)	32	R/W	See section	16.2.11/497
2DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR39)	32	R/W	See section	16.2.11/497
2E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR40)	32	R/W	See section	16.2.11/497
2E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR41)	32	R/W	See section	16.2.11/497
2E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR42)	32	R/W	See section	16.2.11/497
2EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR43)	32	R/W	See section	16.2.11/497
2F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR44)	32	R/W	See section	16.2.11/497
2F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR45)	32	R/W	See section	16.2.11/497
2F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR46)	32	R/W	See section	16.2.11/497

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR47)	32	R/W	See section	16.2.11/497
300	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR48)	32	R/W	See section	16.2.11/497
304	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR49)	32	R/W	See section	16.2.11/497
308	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR50)	32	R/W	See section	16.2.11/497
30C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR51)	32	R/W	See section	16.2.11/497
310	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR52)	32	R/W	See section	16.2.11/497
314	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR53)	32	R/W	See section	16.2.11/497
318	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR54)	32	R/W	See section	16.2.11/497
31C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR55)	32	R/W	See section	16.2.11/497
320	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR56)	32	R/W	See section	16.2.11/497
324	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR57)	32	R/W	See section	16.2.11/497
328	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR58)	32	R/W	See section	16.2.11/497
32C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR59)	32	R/W	See section	16.2.11/497
330	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR60)	32	R/W	See section	16.2.11/497
334	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR61)	32	R/W	See section	16.2.11/497
338	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR62)	32	R/W	See section	16.2.11/497
33C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR63)	32	R/W	See section	16.2.11/497
340	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR64)	32	R/W	See section	16.2.11/497
344	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR65)	32	R/W	See section	16.2.11/497
348	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR66)	32	R/W	See section	16.2.11/497
34C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR67)	32	R/W	See section	16.2.11/497
350	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR68)	32	R/W	See section	16.2.11/497

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
354	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR69)	32	R/W	See section	16.2.11/497
358	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR70)	32	R/W	See section	16.2.11/497
35C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR71)	32	R/W	See section	16.2.11/497
360	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR72)	32	R/W	See section	16.2.11/497
364	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR73)	32	R/W	See section	16.2.11/497
368	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR74)	32	R/W	See section	16.2.11/497
36C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR75)	32	R/W	See section	16.2.11/497
370	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR76)	32	R/W	See section	16.2.11/497
374	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR77)	32	R/W	See section	16.2.11/497
378	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR78)	32	R/W	See section	16.2.11/497
37C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR79)	32	R/W	See section	16.2.11/497
380	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR80)	32	R/W	See section	16.2.11/497
384	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR81)	32	R/W	See section	16.2.11/497
388	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR82)	32	R/W	See section	16.2.11/497
38C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR83)	32	R/W	See section	16.2.11/497
390	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR84)	32	R/W	See section	16.2.11/497
394	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR85)	32	R/W	See section	16.2.11/497
398	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR86)	32	R/W	See section	16.2.11/497
39C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR87)	32	R/W	See section	16.2.11/497
3A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR88)	32	R/W	See section	16.2.11/497
3A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR89)	32	R/W	See section	16.2.11/497
3A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR90)	32	R/W	See section	16.2.11/497

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR91)	32	R/W	See section	16.2.11/497
3B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR92)	32	R/W	See section	16.2.11/497
3B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR93)	32	R/W	See section	16.2.11/497
3B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR94)	32	R/W	See section	16.2.11/497
3BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR95)	32	R/W	See section	16.2.11/497
3C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR96)	32	R/W	See section	16.2.11/497
3C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR97)	32	R/W	See section	16.2.11/497
3C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR98)	32	R/W	See section	16.2.11/497
3CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR99)	32	R/W	See section	16.2.11/497
3D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR100)	32	R/W	See section	16.2.11/497
3D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR101)	32	R/W	See section	16.2.11/497
3D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR102)	32	R/W	See section	16.2.11/497
3DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR103)	32	R/W	See section	16.2.11/497
3E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR104)	32	R/W	See section	16.2.11/497
3E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR105)	32	R/W	See section	16.2.11/497
3E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR106)	32	R/W	See section	16.2.11/497
3EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR107)	32	R/W	See section	16.2.11/497
3F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR108)	32	R/W	See section	16.2.11/497
3F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR109)	32	R/W	See section	16.2.11/497
3F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR110)	32	R/W	See section	16.2.11/497
3FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR111)	32	R/W	See section	16.2.11/497
400	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR112)	32	R/W	See section	16.2.11/497

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
404	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR113)	32	R/W	See section	16.2.11/497
408	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR114)	32	R/W	See section	16.2.11/497
40C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR115)	32	R/W	See section	16.2.11/497
410	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR116)	32	R/W	See section	16.2.11/497
414	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR117)	32	R/W	See section	16.2.11/497
418	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR118)	32	R/W	See section	16.2.11/497
41C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR119)	32	R/W	See section	16.2.11/497
420	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR120)	32	R/W	See section	16.2.11/497
424	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR121)	32	R/W	See section	16.2.11/497
428	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR122)	32	R/W	See section	16.2.11/497
42C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR123)	32	R/W	See section	16.2.11/497
430	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR124)	32	R/W	See section	16.2.11/497
434	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR125)	32	R/W	See section	16.2.11/497
438	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR126)	32	R/W	See section	16.2.11/497
43C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR127)	32	R/W	See section	16.2.11/497
440	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR128)	32	R/W	See section	16.2.11/497
444	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR129)	32	R/W	See section	16.2.11/497
448	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR130)	32	R/W	See section	16.2.11/497
44C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR131)	32	R/W	See section	16.2.11/497
450	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR132)	32	R/W	See section	16.2.11/497
454	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR133)	32	R/W	See section	16.2.11/497
458	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR134)	32	R/W	See section	16.2.11/497

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
45C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR135)	32	R/W	See section	16.2.11/497
460	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR136)	32	R/W	See section	16.2.11/497
464	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR137)	32	R/W	See section	16.2.11/497
468	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR138)	32	R/W	See section	16.2.11/497
46C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR139)	32	R/W	See section	16.2.11/497
470	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR140)	32	R/W	See section	16.2.11/497
474	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR141)	32	R/W	See section	16.2.11/497
478	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR142)	32	R/W	See section	16.2.11/497
47C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR143)	32	R/W	See section	16.2.11/497
480	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR144)	32	R/W	See section	16.2.11/497
484	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR145)	32	R/W	See section	16.2.11/497
488	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR146)	32	R/W	See section	16.2.11/497
48C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR147)	32	R/W	See section	16.2.11/497
490	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR148)	32	R/W	See section	16.2.11/497
494	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR149)	32	R/W	See section	16.2.11/497
498	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR150)	32	R/W	See section	16.2.11/497
49C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR151)	32	R/W	See section	16.2.11/497
4A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR152)	32	R/W	See section	16.2.11/497
4A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR153)	32	R/W	See section	16.2.11/497
4A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR154)	32	R/W	See section	16.2.11/497
AC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR32)	32	R/W	See section	16.2.12/500
AC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR33)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR34)	32	R/W	See section	16.2.12/500
ACC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR35)	32	R/W	See section	16.2.12/500
AD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR36)	32	R/W	See section	16.2.12/500
AD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR37)	32	R/W	See section	16.2.12/500
AD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR38)	32	R/W	See section	16.2.12/500
ADC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR39)	32	R/W	See section	16.2.12/500
AE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR40)	32	R/W	See section	16.2.12/500
AE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR41)	32	R/W	See section	16.2.12/500
AE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR42)	32	R/W	See section	16.2.12/500
AEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR43)	32	R/W	See section	16.2.12/500
AF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR44)	32	R/W	See section	16.2.12/500
AF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR45)	32	R/W	See section	16.2.12/500
AF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR46)	32	R/W	See section	16.2.12/500
AFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR47)	32	R/W	See section	16.2.12/500
B00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR48)	32	R/W	See section	16.2.12/500
B04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR49)	32	R/W	See section	16.2.12/500
B08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR50)	32	R/W	See section	16.2.12/500
B0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR51)	32	R/W	See section	16.2.12/500
B10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR52)	32	R/W	See section	16.2.12/500
B14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR53)	32	R/W	See section	16.2.12/500
B18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR54)	32	R/W	See section	16.2.12/500
B1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR55)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR56)	32	R/W	See section	16.2.12/500
B24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR57)	32	R/W	See section	16.2.12/500
B28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR58)	32	R/W	See section	16.2.12/500
B2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR59)	32	R/W	See section	16.2.12/500
B30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR60)	32	R/W	See section	16.2.12/500
B34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR61)	32	R/W	See section	16.2.12/500
B38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR62)	32	R/W	See section	16.2.12/500
B3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR63)	32	R/W	See section	16.2.12/500
B40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR64)	32	R/W	See section	16.2.12/500
B44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR65)	32	R/W	See section	16.2.12/500
B48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR66)	32	R/W	See section	16.2.12/500
B4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR67)	32	R/W	See section	16.2.12/500
B50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR68)	32	R/W	See section	16.2.12/500
B54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR69)	32	R/W	See section	16.2.12/500
B58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR70)	32	R/W	See section	16.2.12/500
B5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR71)	32	R/W	See section	16.2.12/500
B60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR72)	32	R/W	See section	16.2.12/500
B64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR73)	32	R/W	See section	16.2.12/500
B68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR74)	32	R/W	See section	16.2.12/500
B6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR75)	32	R/W	See section	16.2.12/500
B70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR76)	32	R/W	See section	16.2.12/500
B74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR77)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR78)	32	R/W	See section	16.2.12/500
B7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR79)	32	R/W	See section	16.2.12/500
B80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR80)	32	R/W	See section	16.2.12/500
B84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR81)	32	R/W	See section	16.2.12/500
B88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR82)	32	R/W	See section	16.2.12/500
B8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR83)	32	R/W	See section	16.2.12/500
B90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR84)	32	R/W	See section	16.2.12/500
B94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR85)	32	R/W	See section	16.2.12/500
B98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR86)	32	R/W	See section	16.2.12/500
B9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR87)	32	R/W	See section	16.2.12/500
BA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR88)	32	R/W	See section	16.2.12/500
BA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR89)	32	R/W	See section	16.2.12/500
BA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR90)	32	R/W	See section	16.2.12/500
BAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR91)	32	R/W	See section	16.2.12/500
BB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR92)	32	R/W	See section	16.2.12/500
BB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR93)	32	R/W	See section	16.2.12/500
BB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR94)	32	R/W	See section	16.2.12/500
BBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR95)	32	R/W	See section	16.2.12/500
BC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR96)	32	R/W	See section	16.2.12/500
BC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR97)	32	R/W	See section	16.2.12/500
BC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR98)	32	R/W	See section	16.2.12/500
BCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR99)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR100)	32	R/W	See section	16.2.12/500
BD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR101)	32	R/W	See section	16.2.12/500
BD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR102)	32	R/W	See section	16.2.12/500
BDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR103)	32	R/W	See section	16.2.12/500
BE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR104)	32	R/W	See section	16.2.12/500
BE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR105)	32	R/W	See section	16.2.12/500
BE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR106)	32	R/W	See section	16.2.12/500
BEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR107)	32	R/W	See section	16.2.12/500
BF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR108)	32	R/W	See section	16.2.12/500
BF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR109)	32	R/W	See section	16.2.12/500
BF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR110)	32	R/W	See section	16.2.12/500
BFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR111)	32	R/W	See section	16.2.12/500
C00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR112)	32	R/W	See section	16.2.12/500
C04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR113)	32	R/W	See section	16.2.12/500
C08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR114)	32	R/W	See section	16.2.12/500
C0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR115)	32	R/W	See section	16.2.12/500
C10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR116)	32	R/W	See section	16.2.12/500
C14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR117)	32	R/W	See section	16.2.12/500
C18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR118)	32	R/W	See section	16.2.12/500
C1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR119)	32	R/W	See section	16.2.12/500
C20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR120)	32	R/W	See section	16.2.12/500
C24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR121)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR122)	32	R/W	See section	16.2.12/500
C2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR123)	32	R/W	See section	16.2.12/500
C30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR124)	32	R/W	See section	16.2.12/500
C34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR125)	32	R/W	See section	16.2.12/500
C38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR126)	32	R/W	See section	16.2.12/500
C3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR127)	32	R/W	See section	16.2.12/500
C40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR128)	32	R/W	See section	16.2.12/500
C44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR129)	32	R/W	See section	16.2.12/500
C48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR130)	32	R/W	See section	16.2.12/500
C4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR131)	32	R/W	See section	16.2.12/500
C50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR132)	32	R/W	See section	16.2.12/500
C54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR133)	32	R/W	See section	16.2.12/500
C58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR134)	32	R/W	See section	16.2.12/500
C5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR135)	32	R/W	See section	16.2.12/500
C60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR136)	32	R/W	See section	16.2.12/500
C64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR137)	32	R/W	See section	16.2.12/500
C68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR138)	32	R/W	See section	16.2.12/500
C6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR139)	32	R/W	See section	16.2.12/500
C70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR140)	32	R/W	See section	16.2.12/500
C74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR141)	32	R/W	See section	16.2.12/500
C78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR142)	32	R/W	See section	16.2.12/500
C7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR143)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR144)	32	R/W	See section	16.2.12/500
C84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR145)	32	R/W	See section	16.2.12/500
C88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR146)	32	R/W	See section	16.2.12/500
C8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR147)	32	R/W	See section	16.2.12/500
C90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR148)	32	R/W	See section	16.2.12/500
C94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR149)	32	R/W	See section	16.2.12/500
C98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR150)	32	R/W	See section	16.2.12/500
C9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR151)	32	R/W	See section	16.2.12/500
CA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR152)	32	R/W	See section	16.2.12/500
CA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR153)	32	R/W	See section	16.2.12/500
CA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR154)	32	R/W	See section	16.2.12/500
CAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR155)	32	R/W	See section	16.2.12/500
CB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR156)	32	R/W	See section	16.2.12/500
CB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR157)	32	R/W	See section	16.2.12/500
CB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR158)	32	R/W	See section	16.2.12/500
CBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR159)	32	R/W	See section	16.2.12/500
CC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR160)	32	R/W	See section	16.2.12/500
CC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR161)	32	R/W	See section	16.2.12/500
CC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR162)	32	R/W	See section	16.2.12/500
CCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR163)	32	R/W	See section	16.2.12/500
CD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR164)	32	R/W	See section	16.2.12/500
CD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR165)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR166)	32	R/W	See section	16.2.12/500
CDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR167)	32	R/W	See section	16.2.12/500
CE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR168)	32	R/W	See section	16.2.12/500
CE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR169)	32	R/W	See section	16.2.12/500
CE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR170)	32	R/W	See section	16.2.12/500
CEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR171)	32	R/W	See section	16.2.12/500
CF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR172)	32	R/W	See section	16.2.12/500
CF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR173)	32	R/W	See section	16.2.12/500
CF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR174)	32	R/W	See section	16.2.12/500
CFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR175)	32	R/W	See section	16.2.12/500
D00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR176)	32	R/W	See section	16.2.12/500
D04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR177)	32	R/W	See section	16.2.12/500
D08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR178)	32	R/W	See section	16.2.12/500
D0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR179)	32	R/W	See section	16.2.12/500
D10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR180)	32	R/W	See section	16.2.12/500
D14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR181)	32	R/W	See section	16.2.12/500
D18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR182)	32	R/W	See section	16.2.12/500
D1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR183)	32	R/W	See section	16.2.12/500
D20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR184)	32	R/W	See section	16.2.12/500
D24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR185)	32	R/W	See section	16.2.12/500
D28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR186)	32	R/W	See section	16.2.12/500
D2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR187)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR188)	32	R/W	See section	16.2.12/500
D34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR189)	32	R/W	See section	16.2.12/500
D38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR190)	32	R/W	See section	16.2.12/500
D3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR191)	32	R/W	See section	16.2.12/500
D40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR192)	32	R/W	See section	16.2.12/500
D44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR193)	32	R/W	See section	16.2.12/500
D48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR194)	32	R/W	See section	16.2.12/500
D4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR195)	32	R/W	See section	16.2.12/500
D50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR196)	32	R/W	See section	16.2.12/500
D54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR197)	32	R/W	See section	16.2.12/500
D58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR198)	32	R/W	See section	16.2.12/500
D5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR199)	32	R/W	See section	16.2.12/500
D60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR200)	32	R/W	See section	16.2.12/500
D64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR201)	32	R/W	See section	16.2.12/500
D68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR202)	32	R/W	See section	16.2.12/500
D6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR203)	32	R/W	See section	16.2.12/500
D70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR204)	32	R/W	See section	16.2.12/500
D74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR205)	32	R/W	See section	16.2.12/500
D78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR206)	32	R/W	See section	16.2.12/500
D7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR207)	32	R/W	See section	16.2.12/500
D80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR208)	32	R/W	See section	16.2.12/500
D84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR209)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR210)	32	R/W	See section	16.2.12/500
D8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR211)	32	R/W	See section	16.2.12/500
D90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR212)	32	R/W	See section	16.2.12/500
D94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR213)	32	R/W	See section	16.2.12/500
D98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR214)	32	R/W	See section	16.2.12/500
D9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR215)	32	R/W	See section	16.2.12/500
DA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR216)	32	R/W	See section	16.2.12/500
DA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR217)	32	R/W	See section	16.2.12/500
DA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR218)	32	R/W	See section	16.2.12/500
DAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR219)	32	R/W	See section	16.2.12/500
DB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR220)	32	R/W	See section	16.2.12/500
DB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR221)	32	R/W	See section	16.2.12/500
DB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR222)	32	R/W	See section	16.2.12/500
DBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR223)	32	R/W	See section	16.2.12/500
DC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR224)	32	R/W	See section	16.2.12/500
DC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR225)	32	R/W	See section	16.2.12/500
DC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR226)	32	R/W	See section	16.2.12/500
DCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR227)	32	R/W	See section	16.2.12/500
DD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR228)	32	R/W	See section	16.2.12/500
DD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR229)	32	R/W	See section	16.2.12/500
DD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR230)	32	R/W	See section	16.2.12/500
DDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR231)	32	R/W	See section	16.2.12/500

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR232)	32	R/W	See section	16.2.12/500
DE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR233)	32	R/W	See section	16.2.12/500
1300	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO0_3)	32	R/W	0000_0000h	16.2.13/501
1304	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO4_7)	32	R/W	0000_0000h	16.2.13/501
1308	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO8_11)	32	R/W	0000_0000h	16.2.13/501
130C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO12_15)	32	R/W	0000_0000h	16.2.13/501
1310	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO16_19)	32	R/W	0000_0000h	16.2.13/501
1314	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO20_23)	32	R/W	0000_0000h	16.2.13/501
1318	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO24_27)	32	R/W	0000_0000h	16.2.13/501
131C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO28_31)	32	R/W	0000_0000h	16.2.13/501
1320	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO32_35)	32	R/W	0000_0000h	16.2.13/501
1324	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO36_39)	32	R/W	0000_0000h	16.2.13/501
1328	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO40_43)	32	R/W	0000_0000h	16.2.13/501
132C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO44_47)	32	R/W	0000_0000h	16.2.13/501
1330	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO48_51)	32	R/W	0000_0000h	16.2.13/501
1334	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO52_55)	32	R/W	0000_0000h	16.2.13/501
1338	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO56_59)	32	R/W	0000_0000h	16.2.13/501
133C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO60_63)	32	R/W	0000_0000h	16.2.13/501
1340	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO64_67)	32	R/W	0000_0000h	16.2.13/501
1344	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO68_71)	32	R/W	0000_0000h	16.2.13/501
1348	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO72_75)	32	R/W	0000_0000h	16.2.13/501
134C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO76_79)	32	R/W	0000_0000h	16.2.13/501
1350	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO80_83)	32	R/W	0000_0000h	16.2.13/501
1354	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO84_87)	32	R/W	0000_0000h	16.2.13/501

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1358	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO88_91)	32	R/W	0000_0000h	16.2.13/501
135C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO92_95)	32	R/W	0000_0000h	16.2.13/501
1360	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO96_99)	32	R/W	0000_0000h	16.2.13/501
1364	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO100_103)	32	R/W	0000_0000h	16.2.13/501
1368	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO104_107)	32	R/W	0000_0000h	16.2.13/501
136C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO108_111)	32	R/W	0000_0000h	16.2.13/501
1370	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO112_115)	32	R/W	0000_0000h	16.2.13/501
1374	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO116_119)	32	R/W	0000_0000h	16.2.13/501
1378	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO120_123)	32	R/W	0000_0000h	16.2.13/501
137C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO124_127)	32	R/W	0000_0000h	16.2.13/501
1380	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO128_131)	32	R/W	0000_0000h	16.2.13/501
1384	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO132_135)	32	R/W	0000_0000h	16.2.13/501
1388	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO136_139)	32	R/W	0000_0000h	16.2.13/501
138C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO140_143)	32	R/W	0000_0000h	16.2.13/501
1390	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO144_147)	32	R/W	0000_0000h	16.2.13/501
1394	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO148_151)	32	R/W	0000_0000h	16.2.13/501
1398	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO152_155)	32	R/W	0000_0000h	16.2.13/501
1500	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI0_3)	32	R	0000_0000h	16.2.14/502
1504	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI4_7)	32	R	0000_0000h	16.2.14/502
1508	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI8_11)	32	R	0000_0000h	16.2.14/502
150C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI12_15)	32	R	0000_0000h	16.2.14/502
1510	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI16_19)	32	R	0000_0000h	16.2.14/502
1514	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI20_23)	32	R	0000_0000h	16.2.14/502
1518	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI24_27)	32	R	0000_0000h	16.2.14/502
151C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI28_31)	32	R	0000_0000h	16.2.14/502
1520	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI32_35)	32	R	0000_0000h	16.2.14/502
1524	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI36_39)	32	R	0000_0000h	16.2.14/502

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1528	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI40_43)	32	R	0000_0000h	16.2.14/502
152C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI44_47)	32	R	0000_0000h	16.2.14/502
1530	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI48_51)	32	R	0000_0000h	16.2.14/502
1534	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI52_55)	32	R	0000_0000h	16.2.14/502
1538	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI56_59)	32	R	0000_0000h	16.2.14/502
153C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI60_63)	32	R	0000_0000h	16.2.14/502
1540	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI64_67)	32	R	0000_0000h	16.2.14/502
1544	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI68_71)	32	R	0000_0000h	16.2.14/502
1548	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI72_75)	32	R	0000_0000h	16.2.14/502
154C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI76_79)	32	R	0000_0000h	16.2.14/502
1550	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI80_83)	32	R	0000_0000h	16.2.14/502
1554	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI84_87)	32	R	0000_0000h	16.2.14/502
1558	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI88_91)	32	R	0000_0000h	16.2.14/502
155C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI92_95)	32	R	0000_0000h	16.2.14/502
1560	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI96_99)	32	R	0000_0000h	16.2.14/502
1564	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI100_103)	32	R	0000_0000h	16.2.14/502
1568	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI104_107)	32	R	0000_0000h	16.2.14/502
156C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI108_111)	32	R	0000_0000h	16.2.14/502
1570	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI112_115)	32	R	0000_0000h	16.2.14/502
1574	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI116_119)	32	R	0000_0000h	16.2.14/502
1578	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI120_123)	32	R	0000_0000h	16.2.14/502
157C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI124_127)	32	R	0000_0000h	16.2.14/502
1580	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI128_131)	32	R	0000_0000h	16.2.14/502
1584	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI132_135)	32	R	0000_0000h	16.2.14/502
1588	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI136_139)	32	R	0000_0000h	16.2.14/502
158C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI140_143)	32	R	0000_0000h	16.2.14/502
1590	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI144_147)	32	R	0000_0000h	16.2.14/502
1594	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI148_151)	32	R	0000_0000h	16.2.14/502
1598	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI152_155)	32	R	0000_0000h	16.2.14/502

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1700	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0)	16	R/W	0000h	16.2.15/504
1702	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO1)	16	R/W	0000h	16.2.15/504
1704	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO2)	16	R/W	0000h	16.2.15/504
1706	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO3)	16	R/W	0000h	16.2.15/504
1708	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO4)	16	R/W	0000h	16.2.15/504
170A	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO5)	16	R/W	0000h	16.2.15/504
170C	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO6)	16	R/W	0000h	16.2.15/504
170E	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO7)	16	R/W	0000h	16.2.15/504
1710	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO8)	16	R/W	0000h	16.2.15/504
1712	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO9)	16	R/W	0000h	16.2.15/504
1740	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0)	16	R	0000h	16.2.16/505
1742	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI1)	16	R	0000h	16.2.16/505
1744	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI2)	16	R	0000h	16.2.16/505
1746	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI3)	16	R	0000h	16.2.16/505
1748	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI4)	16	R	0000h	16.2.16/505
174A	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI5)	16	R	0000h	16.2.16/505
174C	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI6)	16	R	0000h	16.2.16/505
174E	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI7)	16	R	0000h	16.2.16/505
1750	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI8)	16	R	0000h	16.2.16/505
1752	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI9)	16	R	0000h	16.2.16/505
1780	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0)	32	W	0000_0000h	16.2.17/505
1784	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO1)	32	W	0000_0000h	16.2.17/505
1788	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO2)	32	W	0000_0000h	16.2.17/505
178C	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO3)	32	W	0000_0000h	16.2.17/505
1790	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO4)	32	W	0000_0000h	16.2.17/505
1794	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO5)	32	W	0000_0000h	16.2.17/505
1798	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO6)	32	W	0000_0000h	16.2.17/505

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
179C	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO7)	32	W	0000_0000h	16.2.17/505
17A0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO8)	32	W	0000_0000h	16.2.17/505
17A4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO9)	32	W	0000_0000h	16.2.17/505

16.2.1 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

This register holds identification information about the device.

NOTE

This register supports only 32-bit accesses. Byte and half-word accesses are not supported.

NOTE

See the chip-specific SIUL2 information for this register's reset value.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ED	PKG					0		MAJOR_MASK				MINOR_MASK			
W																
Reset	0	*	*	*	*	*	0	0	*	*	*	*	*	*	*	*

* Notes:
- MINOR_MASK field: Value is set at the factory and increments with each revision of the device.
- MAJOR_MASK field: Value is set at factory and increments with each revision of the device.
- PKG field: Values are set at factory and cannot be modified.
- PARTNUM field: Value is set at factory and cannot be changed.

SIUL2_MIDR1 field descriptions

Field	Description
0–15 PARTNUM	MCU Part Number For the full part number this field must be combined with MIDR2.PARTNUM[23:16]

Table continues on the next page...

SIUL2_MIDR1 field descriptions (continued)

Field	Description
16 ED	Always reads 0.
17–21 PKG	Package Settings This field can by read by software to determine the package type that is used for the particular device: 0b01101: 144-pin QFP 0b10000: 208-ball BGA All other bit combinations are reserved.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 MAJOR_MASK	Major Mask Revision
28–31 MINOR_MASK	Minor Mask Revision

16.2.2 SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

NOTE

This register supports only 32-bit accesses. Byte and half-word accesses are not supported.

NOTE

See the chip-specific SIUL2 information for this register's reset value.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE_1				FLASH_SIZE_2			0							
W																
Reset	0	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM								0							
W																
Reset	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:

- PARTNUM field: Value is set at factory and cannot be changed.
- FLASH_SIZE_2 field: Value is set at factory and cannot be changed.
- FLASH_SIZE_1 field: Value is set at factory and cannot be changed.

SIUL2_MIDR2 field descriptions

Field	Description
0 SF	Manufacturer 0 Freescale Semiconductor, Inc. 1 Reserved
1–4 FLASH_SIZE_1	Coarse granularity for flash memory size The value of this field needs to be combined with the value of the FLASH_SIZE_2 field to calculate the actual memory size. 0b0110: 1 MB 0b0111: 2 MB This value is set at the factory and cannot be changed. Other values are reserved.
5–8 FLASH_SIZE_2	Fine granularity for flash memory size The value of this field needs to be combined with the value of the FLASH_SIZE_1 field to calculate the actual memory size. 0b0000: 0 x (FLASH_SIZE_1 / 8) 0b0010: 2 x (FLASH_SIZE_1 / 8) 0b0100: 4 x (FLASH_SIZE_1 / 8) This value is set at the factory and cannot be changed.
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 PARTNUM	ASCII character in MCU Part Number This field specifies the part number suffix, and needs to be combined with MIDR1[PARTNUM] to provide the full chip number. 0x50 P This value is set at the factory and cannot be changed. All other values are reserved.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

16.2.3 SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)

The DMA/Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set regardless of the state of the corresponding DMA/Interrupt Request Enable bit in DMA/Interrupt Request Enable Register (SIUL2_DIRER0) - Standard. The IRQ Flag bit remains set until cleared by software or through the servicing of a DMA request. The IRQ Flag bits are cleared by writing a '1' to the bits. A write of '0' has no effect. The number of bits implemented is specified by the parameter that defines the number external IRQ pins.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIF31	EIF30	EIF29	EIF28	EIF27	EIF26	EIF25	EIF24	EIF23	EIF22	EIF21	EIF20	EIF19	EIF18	EIF17	EIF16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DISR0 field descriptions

Field	Description
0 EIF31	External Interrupt Status Flag 31 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER31), EIF31 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER31 and SIUL2_IFEER31 has occurred
1 EIF30	External Interrupt Status Flag 30 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER30), EIF30 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER30 and SIUL2_IFEER30 has occurred
2 EIF29	External Interrupt Status Flag 29 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER29, EIF29 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER29 and SIUL2_IFEER29 has occurred
3 EIF28	External Interrupt Status Flag 28 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER28), EIF28 causes an interrupt or DMA request.

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
	0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER28 and SIUL2_IFEER28 has occurred
4 EIF27	External Interrupt Status Flag 27 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER27), EIF27 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER27 and SIUL2_IFEER27 has occurred
5 EIF26	External Interrupt Status Flag 26 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER26), EIF26 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER26 and SIUL2_IFEER26 has occurred
6 EIF25	External Interrupt Status Flag 25 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER25), EIF25 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER25 and SIUL2_IFEER25 has occurred
7 EIF24	External Interrupt Status Flag 24 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER24), EIF24 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER24 and SIUL2_IFEER24 has occurred
8 EIF23	External Interrupt Status Flag 23 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER23), EIF23 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER23 and SIUL2_IFEER23 has occurred
9 EIF22	External Interrupt Status Flag 22 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER22), EIF22 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER22 and SIUL2_IFEER22 has occurred
10 EIF21	External Interrupt Status Flag 21 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER21), EIF21 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER21 and SIUL2_IFEER21 has occurred
11 EIF20	External Interrupt Status Flag 20

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
	This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER20), EIF20 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER20 and SIUL2_IFEER20 has occurred
12 EIF19	External Interrupt Status Flag 19 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER19), EIF19 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER19 and SIUL2_IFEER19 has occurred
13 EIF18	External Interrupt Status Flag 18 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER18), EIF18 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER18 and SIUL2_IFEER18 has occurred
14 EIF17	External Interrupt Status Flag 17 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER17), EIF17 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER17 and SIUL2_IFEER17 has occurred
15 EIF16	External Interrupt Status Flag 16 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER16), EIF16 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER16 and SIUL2_IFEER16 has occurred
16 EIF15	External Interrupt Status Flag 15 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER15), EIF15 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER15 and SIUL2_IFEER15 has occurred
17 EIF14	External Interrupt Status Flag 14 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER14), EIF14 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER14 and SIUL2_IFEER14 has occurred
18 EIF13	External Interrupt Status Flag 13 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER13), EIF13 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER13 and SIUL2_IFEER13 has occurred

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
19 EIF12	External Interrupt Status Flag 12 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER12), EIF12 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER12 and SIUL2_IFEER12 has occurred
20 EIF11	External Interrupt Status Flag 11 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER11), EIF11 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER11 and SIUL2_IFEER11 has occurred
21 EIF10	External Interrupt Status Flag 10 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER10), EIF10 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER10 and SIUL2_IFEER10 has occurred
22 EIF9	External Interrupt Status Flag 9 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER9), EIF9 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER9 and SIUL2_IFEER9 has occurred
23 EIF8	External Interrupt Status Flag 8 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER8), EIF8 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER8 and SIUL2_IFEER8 has occurred
24 EIF7	External Interrupt Status Flag 7 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER7), EIF7 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER7 and SIUL2_IFEER7 has occurred
25 EIF6	External Interrupt Status Flag 6 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER6), EIF6 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER6 and SIUL2_IFEER6 has occurred
26 EIF5	External Interrupt Status Flag 5 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER5), EIF5 causes an interrupt or DMA request.

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

Field	Description
	0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER5 and SIUL2_IFEER5 has occurred
27 EIF4	External Interrupt Status Flag 4 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER4), EIF4 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER4 and SIUL2_IFEER4 has occurred
28 EIF3	External Interrupt Status Flag 3 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER3), EIF3 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER3 and SIUL2_IFEER3 has occurred
29 EIF2	External Interrupt Status Flag 2 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER2), EIF2 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER2 and SIUL2_IFEER2 has occurred
30 EIF1	External Interrupt Status Flag 1 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER1), EIF1 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER1 and SIUL2_IFEER1 has occurred
31 EIF0	External Interrupt Status Flag 0 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER0), EIF0 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER0 and SIUL2_IFEER0 has occurred

16.2.4 SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)

The DMA/Interrupt Request Enable Register enables the assertion of DMA or interrupt request if the corresponding External IRQ Flag bit is set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0). The number of bits implemented is specified by the parameter that defines the number of external IRQ pins. This register is used to enable the interrupt messaging to the interrupt controller.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EIRE31	EIRE30	EIRE29	EIRE28	EIRE27	EIRE26	EIRE25	EIRE24	EIRE23	EIRE22	EIRE21	EIRE20	EIRE19	EIRE18	EIRE17	EIRE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DIRER0 field descriptions

Field	Description
0 EIRE31	External Interrupt or DMA Request Enable 31 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
1 EIRE30	External Interrupt or DMA Request Enable 30 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
2 EIRE29	External Interrupt or DMA Request Enable 29 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
3 EIRE28	External Interrupt or DMA Request Enable 28 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
4 EIRE27	External Interrupt or DMA Request Enable 27 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
5 EIRE26	External Interrupt or DMA Request Enable 26 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
6 EIRE25	External Interrupt or DMA Request Enable 25 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
7 EIRE24	External Interrupt or DMA Request Enable 24

Table continues on the next page...

SIUL2_DIRER0 field descriptions (continued)

Field	Description
	0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
8 EIRE23	External Interrupt or DMA Request Enable 23 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
9 EIRE22	External Interrupt or DMA Request Enable 22 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
10 EIRE21	External Interrupt or DMA Request Enable 21 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
11 EIRE20	External Interrupt or DMA Request Enable 20 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
12 EIRE19	External Interrupt or DMA Request Enable 19 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
13 EIRE18	External Interrupt or DMA Request Enable 18 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
14 EIRE17	External Interrupt or DMA Request Enable 17 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
15 EIRE16	External Interrupt or DMA Request Enable 16 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
16 EIRE15	External Interrupt or DMA Request Enable 15 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
17 EIRE14	External Interrupt or DMA Request Enable 14 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
18 EIRE13	External Interrupt or DMA Request Enable 13 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
19 EIRE12	External Interrupt or DMA Request Enable 12

Table continues on the next page...

SIUL2_DIRER0 field descriptions (continued)

Field	Description
	0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
20 EIRE11	External Interrupt or DMA Request Enable 11 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
21 EIRE10	External Interrupt or DMA Request Enable 10 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
22 EIRE9	External Interrupt or DMA Request Enable 9 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
23 EIRE8	External Interrupt or DMA Request Enable 8 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
24 EIRE7	External Interrupt or DMA Request Enable 7 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
25 EIRE6	External Interrupt or DMA Request Enable 6 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
26 EIRE5	External Interrupt or DMA Request Enable 5 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
27 EIRE4	External Interrupt or DMA Request Enable 4 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
28 EIRE3	External Interrupt or DMA Request Enable 3 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
29 EIRE2	External Interrupt or DMA Request Enable 2 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
30 EIRE1	External Interrupt or DMA Request Enable 1 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
31 EIRE0	External Interrupt or DMA Request Enable 0

Table continues on the next page...

SIUL2_DIRER0 field descriptions (continued)

Field	Description
	0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register

16.2.5 SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)

The DIRSR selects between the DMA or interrupt request. If the corresponding bits are set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0) and SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0), then the DMA/Interrupt Request Select bit determines whether DMA or an interrupt request is asserted. The number of bits implemented is specified by the parameter that defines the number of DMA request outputs.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DIRSR31	DIRSR30	DIRSR29	DIRSR28	DIRSR27	DIRSR26	DIRSR25	DIRSR24	DIRSR23	DIRSR22	DIRSR21	DIRSR20	DIRSR19	DIRSR18	DIRSR17	DIRSR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DIRSR15	DIRSR14	DIRSR13	DIRSR12	DIRSR11	DIRSR10	DIRSR9	DIRSR8	DIRSR7	DIRSR6	DIRSR5	DIRSR4	DIRSR3	DIRSR2	DIRSR1	DIRSR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DIRSR0 field descriptions

Field	Description
0 DIRSR31	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
1 DIRSR30	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved

Table continues on the next page...

SIUL2_DIRSR0 field descriptions (continued)

Field	Description
2 DIRSR29	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
3 DIRSR28	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
4 DIRSR27	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
5 DIRSR26	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
6 DIRSR25	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
7 DIRSR24	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
8 DIRSR23	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
9 DIRSR22	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
10 DIRSR21	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
11 DIRSR20	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved

Table continues on the next page...

SIUL2_DIRSR0 field descriptions (continued)

Field	Description
12 DIRSR19	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
13 DIRSR18	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
14 DIRSR17	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
15 DIRSR16	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
16 DIRSR15	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
17 DIRSR14	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
18 DIRSR13	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
19 DIRSR12	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
20 DIRSR11	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
21 DIRSR10	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved

Table continues on the next page...

SIUL2_DIRSR0 field descriptions (continued)

Field	Description
22 DIRSR9	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
23 DIRSR8	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
24 DIRSR7	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
25 DIRSR6	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
26 DIRSR5	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
27 DIRSR4	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Reserved
28 DIRSR3	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
29 DIRSR2	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
30 DIRSR1	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
31 DIRSR0	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected

16.2.6 SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IREE31	IREE30	IREE29	IREE28	IREE27	IREE26	IREE25	IREE24	IREE23	IREE22	IREE21	IREE20	IREE19	IREE18	IREE17	IREE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IREE15	IREE14	IREE13	IREE12	IREE11	IREE10	IREE9	IREE8	IREE7	IREE6	IREE5	IREE4	IREE3	IREE2	IREE1	IREE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IREER0 field descriptions

Field	Description
0 IREE31	Enable rising-edge events to cause the EIF31 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
1 IREE30	Enable rising-edge events to cause the EIF30 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
2 IREE29	Enable rising-edge events to cause the EIF29 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
3 IREE28	Enable rising-edge events to cause the EIF28 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
4 IREE27	Enable rising-edge events to cause the EIF27 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
5 IREE26	Enable rising-edge events to cause the EIF26 bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
6 IREE25	Enable rising-edge events to cause the EIF25 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
7 IREE24	Enable rising-edge events to cause the EIF24 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
8 IREE23	Enable rising-edge events to cause the EIF23 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
9 IREE22	Enable rising-edge events to cause the EIF22 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
10 IREE21	Enable rising-edge events to cause the EIF21 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
11 IREE20	Enable rising-edge events to cause the EIF20 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
12 IREE19	Enable rising-edge events to cause the EIF19 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
13 IREE18	Enable rising-edge events to cause the EIF18 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
14 IREE17	Enable rising-edge events to cause the EIF17 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
15 IREE16	Enable rising-edge events to cause the EIF16 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
16 IREE15	Enable rising-edge events to cause the EIF15 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
17 IREE14	Enable rising-edge events to cause the EIF14 bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
18 IREE13	Enable rising-edge events to cause the EIF13 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
19 IREE12	Enable rising-edge events to cause the EIF12 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
20 IREE11	Enable rising-edge events to cause the EIF11 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
21 IREE10	Enable rising-edge events to cause the EIF10 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
22 IREE9	Enable rising-edge events to cause the EIF9 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
23 IREE8	Enable rising-edge events to cause the EIF8 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
24 IREE7	Enable rising-edge events to cause the EIF7 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
25 IREE6	Enable rising-edge events to cause the EIF6 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
26 IREE5	Enable rising-edge events to cause the EIF5 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
27 IREE4	Enable rising-edge events to cause the EIF4 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
28 IREE3	Enable rising-edge events to cause the EIF3 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
29 IREE2	Enable rising-edge events to cause the EIF2 bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
30 IREE1	Enable rising-edge events to cause the EIF1 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
31 IREE0	Enable rising-edge events to cause the EIF0 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

16.2.7 SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

This register supports 8-, 16-, and 32-bit accesses.

NOTE

If both the IREE it and IFEE bits are cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IFEE31	IFEE30	IFEE29	IFEE28	IFEE27	IFEE26	IFEE25	IFEE24	IFEE23	IFEE22	IFEE21	IFEE20	IFEE19	IFEE18	IFEE17	IFEE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFEE15	IFEE14	IFEE13	IFEE12	IFEE11	IFEE10	IFEE9	IFEE8	IFEE7	IFEE6	IFEE5	IFEE4	IFEE3	IFEE2	IFEE1	IFEE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFEER0 field descriptions

Field	Description
0 IFEE31	Enable falling-edge events to cause the EIF31 bit to be set.

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
	0 Falling-edge event is disabled 1 Falling-edge event is enabled
1 IFEE30	Enable falling-edge events to cause the EIF30 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
2 IFEE29	Enable falling-edge events to cause the EIF29 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
3 IFEE28	Enable falling-edge events to cause the EIF28 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
4 IFEE27	Enable falling-edge events to cause the EIF27 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
5 IFEE26	Enable falling-edge events to cause the EIF26 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
6 IFEE25	Enable falling-edge events to cause the EIF25 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
7 IFEE24	Enable falling-edge events to cause the EIF24 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
8 IFEE23	Enable falling-edge events to cause the EIF23 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
9 IFEE22	Enable falling-edge events to cause the EIF22 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
10 IFEE21	Enable falling-edge events to cause the EIF21 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
11 IFEE20	Enable falling-edge events to cause the EIF20 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
12 IFEE19	Enable falling-edge events to cause the EIF19 bit to be set.

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
	0 Falling-edge event is disabled 1 Falling-edge event is enabled
13 IFEE18	Enable falling-edge events to cause the EIF18 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
14 IFEE17	Enable falling-edge events to cause the EIF17 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
15 IFEE16	Enable falling-edge events to cause the EIF16 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
16 IFEE15	Enable falling-edge events to cause the EIF15 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
17 IFEE14	Enable falling-edge events to cause the EIF14 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
18 IFEE13	Enable falling-edge events to cause the EIF13 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
19 IFEE12	Enable falling-edge events to cause the EIF12 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
20 IFEE11	Enable falling-edge events to cause the EIF11 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
21 IFEE10	Enable falling-edge events to cause the EIF10 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
22 IFEE9	Enable falling-edge events to cause the EIF9 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
23 IFEE8	Enable falling-edge events to cause the EIF8 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
24 IFEE7	Enable falling-edge events to cause the EIF7 bit to be set.

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
	0 Falling-edge event is disabled 1 Falling-edge event is enabled
25 IFEE6	Enable falling-edge events to cause the EIF6 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
26 IFEE5	Enable falling-edge events to cause the EIF5 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
27 IFEE4	Enable falling-edge events to cause the EIF4 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
28 IFEE3	Enable falling-edge events to cause the EIF3 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
29 IFEE2	Enable falling-edge events to cause the EIF2 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
30 IFEE1	Enable falling-edge events to cause the EIF1 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
31 IFEE0	Enable falling-edge events to cause the EIF0 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

16.2.8 SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs. The number of interrupts supporting this feature is MCU dependent and can be configured between 1 and 32.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IFE31	IFE30	IFE29	IFE28	IFE27	IFE26	IFE25	IFE24	IFE23	IFE22	IFE21	IFE20	IFE19	IFE18	IFE17	IFE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFE15	IFE14	IFE13	IFE12	IFE11	IFE10	IFE9	IFE8	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFER0 field descriptions

Field	Description
0 IFE31	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
1 IFE30	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
2 IFE29	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
3 IFE28	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
4 IFE27	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
5 IFE26	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
6 IFE25	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
7 IFE24	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
8 IFE23	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
9 IFE22	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

Table continues on the next page...

SIUL2_IFER0 field descriptions (continued)

Field	Description
10 IFE21	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
11 IFE20	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
12 IFE19	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
13 IFE18	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
14 IFE17	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
15 IFE16	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
16 IFE15	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
17 IFE14	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
18 IFE13	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
19 IFE12	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
20 IFE11	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
21 IFE10	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

Table continues on the next page...

SIUL2_IFER0 field descriptions (continued)

Field	Description
22 IFE9	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
23 IFE8	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
24 IFE7	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
25 IFE6	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
26 IFE5	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
27 IFE4	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
28 IFE3	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
29 IFE2	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
30 IFE1	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
31 IFE0	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

16.2.9 SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

These registers are used to configure the filter counter associated with each digital glitch filter.

NOTE

> These registers support only 32-bit accesses. Byte and half-word accesses are not supported.

Address: 0h base + 40h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												MAXCNT			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFMCR*n* field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MAXCNT	Maximum Interrupt Filter Counter setting MAXCNT can be 0d to 15d. • For a MAXCNT value of 0d, 1d, or 2d, the filter behaves as an ALL PASS filter. • For a MAXCNT value of 3d to 15d, the filter period is TCK*MAXCNT+n*TCK, where n is 0, 1, 2, 3, or 4. TCK is the prescaled filter clock period, which is the IRC clock prescaled to the IFCP value, which is specified in SIUL2_IFCPR. TIRC is the basic filter clock period: 62.5 ns (f = 16 MHz). **NOTE:** The filter delay is 2 TCK clock cycles more than the filter period. In general, TFILTER and TDELAY are not integer multiples of TCK because DIN is asynchronous whereas DOUT is synchronous with respect to the CK clock. The factor n accounts for this uncertainty in filter period calculation.

16.2.10 SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

This register is used to configure a clock prescaler which is used to select the clock for all digital filters. A prescaler is applied to the input clock to the SIUL2, which is the system AIPS clock counter in the SIUL2.

NOTE

> This register supports only 32-bit accesses. Byte and half-word accesses are not supported.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												IFCP			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFCPR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15

16.2.11 SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR*n*)

These registers select which source signal is connected to the register's associated destination, which is a chip pin that is or can be configured as an output.

For the associated chip-pin destination, the register also specifies the pin's electrical properties.

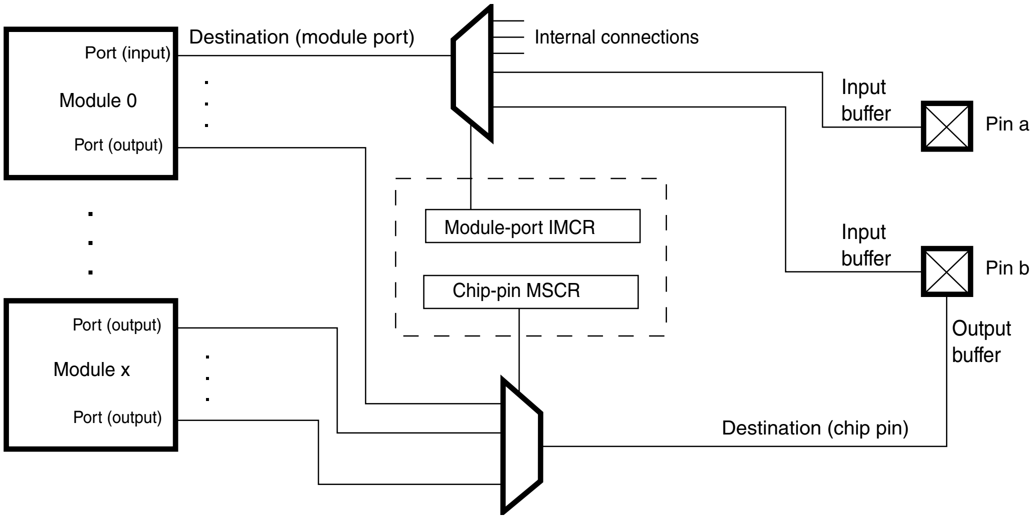


Figure 16-2. MSCR pin connection

The fields in an MSCR vary depending on its associated destination (chip pin).

For chip-pin MSCR assignments, pin types, APC support, and SSS values, see the pin-description table for your chip.

NOTE

This register supports only 32-bit accesses. Byte and half-word write accesses are not supported.

Address: 0h base + 240h offset + (4d × i), where i=0d to 154d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		SRC[1:0]		0		OBE	ODE	SMC	APC	0		IBE	HYS	PUS	PUE
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0							0				SSS			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See the Signal Description IOMUX details for the reset values of the MSCRs.

SIUL2_MSCR*n* field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 SRC[1:0]	Slew Rate Control Used only when the associated destination is a chip pin. 00 Half drive strength with slew rate control 01 Full drive strength with slew rate control 10 Half drive strength without slew rate control 11 Full drive strength without slew rate control
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 OBE	GPIO Output Buffer Enable Applies only to digital pins. Otherwise this bit is reserved. 0 Output driver disabled 1 Output driver enabled
7 ODE	Open Drain Enable **NOTE:** To enable open drain both OBE and ODE bits need to be set. 0 Open drain function disabled 1 Open drain function enabled when OBE is also 1
8 SMC	Safe Mode Control Used only when the associated destination is a chip pin. Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode. 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Don't disable
9 APC	Analog Pad Control Used only when the associated destination is a chip pin that supports analog I/O. Enables the pin's analog-input-path switch.

Table continues on the next page...

SIUL2_MSCR*n* field descriptions (continued)

Field	Description
	0 Disable (the switch is off) 1 Enable (another module can control the state of the switch)
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 IBE	Input Buffer Enable Used only when the associated destination is a chip pin. Enables the associated pin's input buffer. 0 Disabled 1 Enabled
13 HYS	Input Hysteresis Used only when the associated destination is a chip pin. Enables input hysteresis for the associated pin. 0 Disabled 1 Enabled
14 PUS	Pull Select Determines whether the pull function is a pullup or pulldown when the pull function is enabled by the PUE field. Used only when the associated destination is a chip pin. 0 Pulldown 1 Pullup
15 PUE	Pull Enable Enables the pull function. Used only when the associated destination is a chip pin. 0 Disabled 1 Enabled
16 INV	Invert Inverts the signal selected by SSS before transmitting it to the associated destination (chip pin or module port). 0 Don't invert 1 Invert
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin). This field has a variable width depending on the number of alternate functions available for that pin. For a chip pin, the source signals are outputs from module ports.

16.2.12 SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR*n*)

Selects which source signal is connected to the register's associated destination, which is an internal module port that is or can be configured as an input.

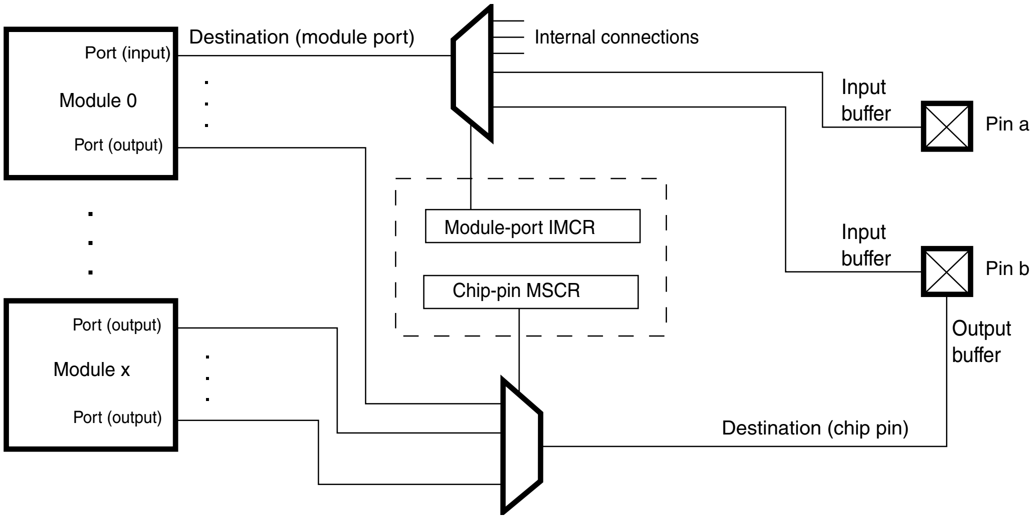


Figure 16-3. IMCR module-port connection

The fields in an IMCR vary depending on its associated destination (module port).

For IMCR assignments and SSS values, see the pin-description table for your chip.

NOTE

This register supports only 32-bit accesses. Byte and half-word write accesses are not supported.

Address: 0h base + AC0h offset + (4d × i), where i=0d to 201d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0							0				SSS			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- See the Signal Description details for the reset values of the IMCRs.

SIUL2_IMCR*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 INV	Invert Inverts the signal selected by SSS before transmitting it to the associated destination (chip pin or module port). 0 Don't invert 1 Invert
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (module port). This field has a variable width depending on the number of alternate functions available for that port. For a module port, the source signals are either outputs from module ports or inputs from chip pins.

16.2.13 SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO*n*)

These registers can be used to set or clear a single GPIO pad with a byte access.

These registers support 8-, 16-, and 32-bit accesses.

Address: 0h base + 1300h offset + (4d × i), where i=0d to 38d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							PDO_4n	0							PDO_4n1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							PDO_4n2	0							PDO_4n3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_GPDO*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIUL2_GPDO*n* field descriptions (continued)

Field	Description
7 PDO_4n	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n represents PDO[4n], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n represents PDO12. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PDO_4n1	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n1 represents PDO[4n+1], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n1 represents PDO13. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 PDO_4n2	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n2 represents PDO[4n+2], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n2 represents PDO14. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PDO_4n3	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n3 represents PDO[4n+3], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n3 represents PDO15. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

16.2.14 SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI*n*)

These registers can be used to read the GPIO pad data with a byte access.

These registers support 8-, 16-, and 32-bit accesses.

Address: 0h base + 1500h offset + (4d × i), where i=0d to 38d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							PDI_4n	0							PDI_4n1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							PDI_4n2	0							PDI_4n3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_GPDI*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDI_4n	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n represents PDI[4n], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n represents PDI12. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PDI_4n1	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n1 represents PDI[4n+1], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n1 represents PDI13. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 PDI_4n2	Pad Data In

Table continues on the next page...

SIUL2_GPDI*n* field descriptions (continued)

Field	Description
	This bit stores the value of the external GPIO pad associated with this register. PDI_4n2 represents PDI[4n+2], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n2 represents PDI14. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PDI_4n3	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n3 represents PDI[4n+3], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n3 represents PDI15. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

16.2.15 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write.

The SIUL2_PGPDO registers access the same physical resource as the SIUL2_PDO and SIUL2_MPGPDO address locations. Some examples of the mapping:

- PPDO[0][0] = PDO[0]
- PPDO[2][0] = PDO[32]
- PPDO[31][15] = PDO[511]

These registers support 8-, 16-, and 32-bit accesses.

Address: 0h base + 1700h offset + (2d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	PPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDO*n* field descriptions

Field	Description
0–15 PPDO	Parallel Pad Data Out Write or read the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO).The x and bit index define which PPDO register bit is equivalent to which PDO register bit according to the following equation: PPDO[x][y] = PDO[(x*16)+y]

16.2.16 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration.The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read.

These registers support 8-, 16-, and 32-bit accesses.

Address: 0h base + 1740h offset + (2d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	PPDI															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDI*n* field descriptions

Field	Description
0–15 PPDI	Parallel Pad Data In Reads the current pad value. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and bit index define which PPDI register bit is equivalent to which PDI register bit according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

16.2.17 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

This register can be used to selectively modify the pad values associated to PPDO[x] [0:15]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read access will return 0.

NOTE

This register supports only 32-bit accesses. Byte and half-word accesses are not supported.

Address: 0h base + 1780h offset + (4d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																0															
W	MASK																MPPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_MPGPDO*n* field descriptions

Field	Description
0–15 MASK	Mask Field Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0b The associated bit value in the MPPDO[x] field is ignored 1b The associated bit value in the MPPDO[x] field is written
16–31 MPPDO	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO).The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][y] = PDO[(x*16)+y]

16.3 Functional description

16.3.1 General

This section provides a complete functional description of the System Integration Unit Lite2.

16.3.2 Pad Control

The SIUL2 is capable of controlling the electrical characteristics of as many as 512 pads. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2. Not all of these features are supported on all devices or on all pads. Support is dependent on the specific application needs and the pads implemented on the device.

The required controls are:

- Slew rate control
  - Offers improved EMC performance
  - Support for a fast and a slow slew rate
- Drive strength control
  - Offers improved EMC performance
  - As many as four different drive strength levels can be supported.
- Pull capability
  - Offers flexibility in the device configuration and the elimination of external hardware in some cases
  - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down, or no-pull enabled
- Input and output buffer control
- Hysteresis
- Open drain
  - Needed to support different pads muxed (for example, to mux IIC with non-open drain pads)
- Pad output assignment
  - Defines which chip function has control over the output of the pad

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for you to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads' individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register, allowing simplified duplication of software for each pad with indexed changes for each pad.

16.3.3 General Purpose Input and Output pads (GPIO)

The SIUL2 allows each pad to be configured as either a General Purpose Input or Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that uses the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage as many as 512 GPIO pads organized as ports that can be accessed for data reads and writes as 8-bit, 16-bi,t or 32-bit.

As shown in the following figure, all port accesses are identical, with each read or write being performed only at a different location to access a different port width.

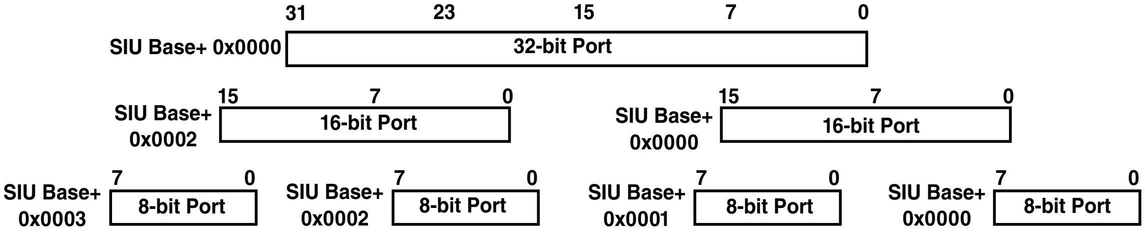


Figure 16-4. Data Port example arrangement showing configuration for different port width accesses

This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations.

The data input registers support read access only.

When a pad is configured to use one of its alternate functions, the data input values reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non-GPIO), this write will not be reflected by the pad value until re-configured to GPIO. All general purpose pads are implemented as bidirectional.

Note

In case the bi-directional operation impacts performance (for example, ADC accuracy) or is not required for a specific pad function, you can limit the functionality of the pad to "Input Only."

16.3.4 External interrupts/DMA requests (REQ Pins)

The SIUL2 supports 1 to 32 external interrupts, which can be allocated to any pad necessary at the MCU level. This allocation is fixed per MCU.

For the assignments of the external interrupts, see the Signal Description details.

The SIUL2 supports one to four interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support as many as eight external interrupt sources from the device pads.

For devices with a number of interrupt sources that is smaller than the maximum number, the MCU can implement either all four interrupt vectors with fewer sources or reduce the number of vectors as the number of sources are reduced. The following figure shows two sample scenarios.

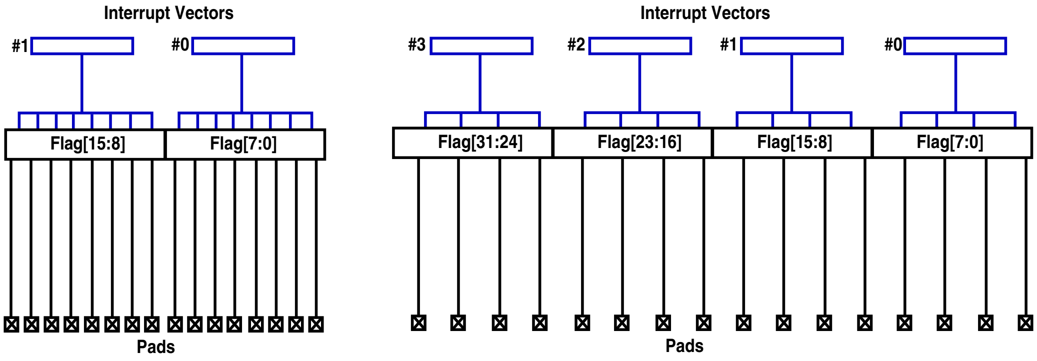


Figure 16-5. Example of different interrupt to vector mapping at MCU level

All the external interrupt pads within a single group (maximum of eight pads) have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for the application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

An MCU-specific number of external interrupt lines can have digital glitch filters applied to them. The supported range is 1 to 32. The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on an interrupt line.

16.3.4.1 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization, the user must do the following:

- Mask interrupts by clearing the EIRE*n* bits in DIRER0
- Select the pin polarity by setting the appropriate IREE*n* bits in IREER0 and the appropriate IFEE*n* bits in IFEER0 as desired
- Configure the appropriate bits in the MSCR[0:511] register for the external interrupt pin(s) desired as follows:
  - Clear the OBE and ODE bits to disable output
  - Set the IBE bit to enable the pin's input buffer
  - If using the internal pullup/pulldown, configure the appropriate PUE and PUS fields

Note

> MSCR[SSS] bits do not need to be changed for either MSCR[0:511] or IMCR[0:511] because the external interrupt pin input is directly connected to the SIUL for EIRQ pins.
>
> External interrupt pins should never be configured as outputs (MSCR[ODC] bits are not zeros) when external interrupt inputs are desired because false interrupts could be detected (such as from a GPIO configuration).

- Select the request desired between DMA or Interrupt by writing the appropriate DIRS*n* bits in DIRSR0
- Select the desired glitch filter setup for the pins by writing the following:
  - Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMC*n* register for the respective external interrupt that is being used

  - Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR register
  - Enable the glitch filter for the desired external interrupt pins by setting the appropriate IFE*n* bits in IFER0
- Write to EIF*n* bits in DISR0 to as desired to clear any flags
- Enable the interrupt pins by setting the appropriate EIRE*n* bits in DIRER0

16.3.4.2 External interrupt management

The user can enable or disable each interrupt independently using a single rolled up register, SIUL2_DIRER0.

A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge, or both edges being active. A setting of having both edge events disabled is reserved and should not be configured. The user controls the active IRQ edge through the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (DISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

Refer to the following figure for an overview of the external interrupt implementation.

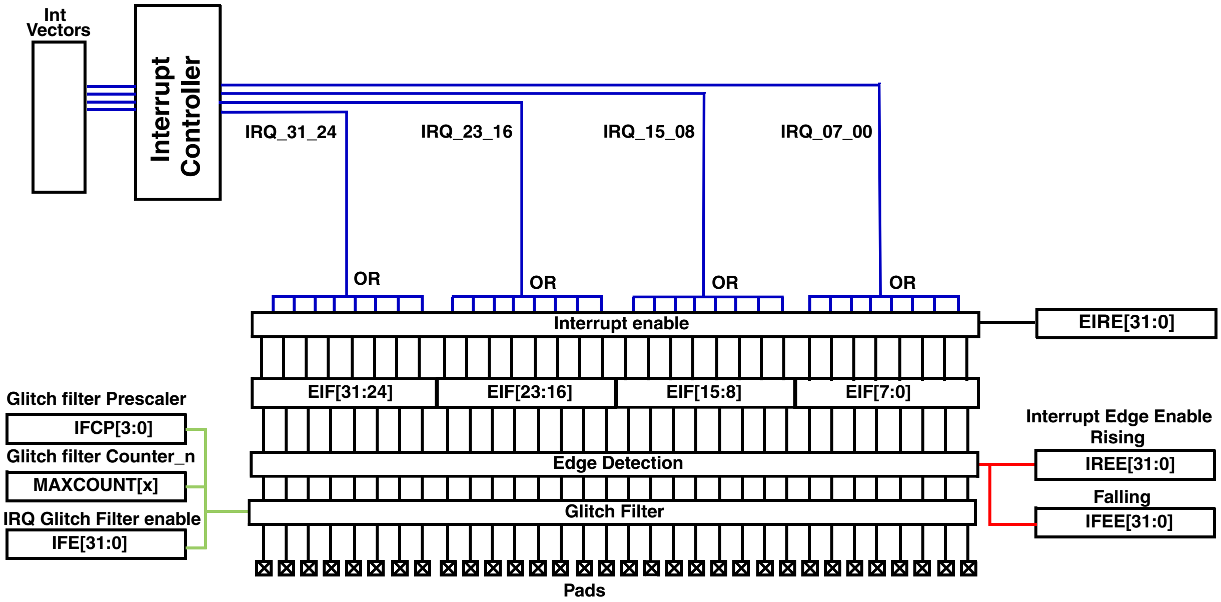


Figure 16-6. External interrupt pad diagram

16.3.4.3 External interrupt request

The REQ input pins on the device are sources for interrupt or DMA requests. The system has four possible interrupt vectors available for the REQ pins in the SIUL2. The mapping of 32 interrupt request sources to vectors and channels appears in the following table.

Table 16-1. Interrupt source mapping to SIUL2 Interrupt request output for 32 interrupt sources

Vector/Channel #	Interrupt vector source
0	REQ[07] I REQ[06] I REQ[05] I REQ[04] I REQ[03] I REQ[02] I REQ[01] I REQ[00]
1	REQ[15] I REQ[14] I REQ[13] I REQ[12] I REQ[11] I REQ[10] I REQ[09] I REQ[08]
2	REQ[23] I REQ[22] I REQ[21] I REQ[20] I REQ[19] I REQ[18] I REQ[17] I REQ[16]
3	REQ[31] I REQ[30] I REQ[29] I REQ[28] I REQ[27] I REQ[26] I REQ[25] I REQ[24]

16.3.4.4 DMA requests

The REQ pins on the device map to independent DMA request channels in the DMA controller. The maximum four pins map corresponding to four DMA request channels. See the DMA request mapping details in the Device Configuration information.

Chapter 17
Crossbar Switch (XBAR)

17.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The information found here provides information on the layout, configuration, and programming of the crossbar switch.

The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows all bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

17.1.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
    - Allows concurrent accesses from different masters to different slaves
    - Slave arbitration attributes configured on a slave-by-slave basis
- Up to single-clock 64-bit transfer
- Support for burst transfers of up to 16 beats of data
- Low-Power Park mode support
- Dynamic master priority elevation

17.2 Memory Map / Register Definition

Each slave port of the crossbar switch contains configuration registers. Read- and write-transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The CRS*n* and PRS*n* registers can be programmed to be read-only to prevent changes to their configuration. After being read-only protected, future writes to them will terminate with an error.

NOTE

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the chip configuration details for the exact master/slave assignments for your device.

All references to the crossbar switch registers are based on the physical port connections.

XBAR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Priority Registers Slave (XBAR_PRS0)	32	R/W	See section	17.2.1/515
10	Control Register (XBAR_CRS0)	32	R/W	See section	17.2.2/517
100	Priority Registers Slave (XBAR_PRS1)	32	R/W	See section	17.2.1/515
110	Control Register (XBAR_CRS1)	32	R/W	See section	17.2.2/517
200	Priority Registers Slave (XBAR_PRS2)	32	R/W	See section	17.2.1/515
210	Control Register (XBAR_CRS2)	32	R/W	See section	17.2.2/517
300	Priority Registers Slave (XBAR_PRS3)	32	R/W	See section	17.2.1/515
310	Control Register (XBAR_CRS3)	32	R/W	See section	17.2.2/517
400	Priority Registers Slave (XBAR_PRS4)	32	R/W	See section	17.2.1/515
410	Control Register (XBAR_CRS4)	32	R/W	See section	17.2.2/517
500	Priority Registers Slave (XBAR_PRS5)	32	R/W	See section	17.2.1/515
510	Control Register (XBAR_CRS5)	32	R/W	See section	17.2.2/517
600	Priority Registers Slave (XBAR_PRS6)	32	R/W	See section	17.2.1/515
610	Control Register (XBAR_CRS6)	32	R/W	See section	17.2.2/517

Table continues on the next page...

XBAR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
700	Priority Registers Slave (XBAR_PRS7)	32	R/W	See section	17.2.1/515
710	Control Register (XBAR_CRS7)	32	R/W	See section	17.2.2/517

17.2.1 Priority Registers Slave (XBAR_PRS*n*)

The priority registers (PRSn) set the priority of each master port on a per slave port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. After the CRSn[RO] bit is set, the PRSn register can only be read; attempts to write to it have no effect on PRSn and result in a bus-error response to the master initiating the write.

Two available masters must not be programmed with the same priority level. Attempts to program two or more masters with the same priority level result in a bus-error response and the PRSn is not updated.

NOTE

Valid values for the M*n* priority fields depend on which masters are available on the chip. This information can be found in the chip-specific information for the crossbar.

- If the chip contains less than five masters, values 0 to 3 are valid. Writing other values will result in an error.
- If the chip contains five or more masters, valid values are 0 to n-1, where n is the number of masters attached to the Crossbar Switch. Other values will result in an error.

NOTE

See the chip-specific crossbar information for the reset value of this register.

NOTE

On the MPC5744P, the reset value of these registers is 0x0320_0010.

Address: 0h base + 0h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	Reserved			0	M6			0	M5			0	Reserved		
W		—												—		
Reset	0	0	0	0	0	0*	0*	0*	0	0*	0*	0*	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	Reserved			0	Reserved			0	M1			0	M0		
W		—				—										
Reset	0	0	0	0	0	0	0	0	0	0*	0*	0*	0	0*	0*	0*

* Notes:

- See the chip-specific crossbar information for the reset value of this register.

XBAR_PRS*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 Reserved	This field is reserved.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 M6	Master 6 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 M5	Master 5 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 Reserved	This field is reserved.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 Reserved	This field is reserved.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

XBAR_PRS*n* field descriptions (continued)

Field	Description
21–23 Reserved	This field is reserved.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 M1	Master 1 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 M0	Master 0 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.

17.2.2 Control Register (XBAR_CRS*n*)

These registers control several features of each slave port and must be accessed using 32-bit accesses. After CRSn[RO] is set, the PRSn can only be read; attempts to write to it have no effect and result in an error response.

NOTE

> See the chip-specific crossbar information for the reset value of this register.
>
> Not all HPE*n* fields may be active. See the chip-specific crossbar information for which masters support high priority elevation. Setting a field corresponding to a master that does not support high-priority elevation has no effect.

Address: 0h base + 10h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	HLP	0						0	HPE6	HPE5	0	0	0	HPE1	HPE0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						ARB		0		PCTL		0	PARK		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- See the chip-specific crossbar information for the reset value of this register.

XBAR_CRS*n* field descriptions

Field	Description
0 RO	Read Only Forces the PRS*n* and CRS*n* registers to be read-only. After being set, only a hardware reset clears this field. 0 The CRS*n* and PRS*n* registers are writeable 1 The CRS*n* and PRS*n* registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HLP	Halt Low Priority Sets the initial arbitration priority for low power mode requests . Setting this bit will not affect the request for low power mode from attaining highest priority once it has control of the slave ports. 0 The low power mode request has the highest priority for arbitration on this slave port 1 The low power mode request has the lowest initial priority for arbitration on this slave port
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 HPE6	On this slave port, enable priority elevation for master 6. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 6 is disabled on this slave port 1 Priority elevation for master 6 is enabled on this slave port
10 HPE5	On this slave port, enable priority elevation for master 5. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 5 is disabled on this slave port 1 Priority elevation for master 5 is enabled on this slave port
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

XBAR_CRS*n* field descriptions (continued)

Field	Description
14 HPE1	On this slave port, enable priority elevation for master 1. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 1 is disabled on this slave port 1 Priority elevation for master 1 is enabled on this slave port
15 HPE0	On this slave port, enable priority elevation for master 0. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 0 is disabled on this slave port 1 Priority elevation for master 0 is enabled on this slave port
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ARB	Arbitration Mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority 10 Reserved 11 Reserved
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PCTL	Parking Control Determines the slave port’s parking control. The low-power park feature results in an overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port 10 Low-power park. When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state 11 Reserved
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. **NOTE:** Select only master ports that are present on the chip. Otherwise, undefined behavior might occur. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5

Table continues on the next page...

XBAR_CRS*n* field descriptions (continued)

Field	Description
	110 Park on master port M6 111 Park on master port M7

17.3 Functional Description

17.3.1 General operation

When a master accesses the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
  - An outstanding request to one slave port that has a long response time and
  - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port. However, if the master is running a fixed-length burst transfer it retains control of the slave port until that transfer completes.

The crossbar terminates all master IDLE transfers, as opposed to allowing the termination to come from one of the slave buses. Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by CRS*n*[PARK] . This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port. The slave port can also be put into Low Power Park mode to save power, by using CRS*n*[PCTL].

17.3.2 Register coherency

The operation of the crossbar is affected as soon as a register is written. The values of the registers do not track with slave-port-related master accesses, but instead track only with slave accesses.

17.3.3 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The arbitration scheme is independently programmable for each slave port.

17.3.3.1 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the priority registers (PRS*n*). If two masters request access to the same slave port, the master with the highest priority in the selected priority register gains control over the slave port.

NOTE

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

Table 17-1. How the Crossbar Switch grants control of a slave port to a master

When	Then the Crossbar Switch grants control to the requesting master
Both of the following are true: • The current master is not running a transfer. • The new requesting master's priority level is higher than that of the current master.	At the next clock edge
Both of the following are true: • The current master is running a fixed length burst transfer or a locked transfer. • The requesting master's priority level is higher than that of the current master.	At the end of the burst transfer or locked transfer
The requesting master's priority level is lower than the current master.	At the conclusion of one of the following cycles: • An IDLE cycle • A non-IDLE cycle to a location other than the current slave port

17.3.3.2 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary, accounting for locked and fixed-length burst transfers. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

Parking may continue to be used in a round-robin mode, but does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

17.3.3.3 Priority assignment

Each master port must be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

17.4 Initialization/application information

No initialization is required for the crossbar switch.

Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. However, settings and priorities may be programmed to achieve maximum system performance.

Chapter 18
Crossbar Integrity Checker (XBIC)

18.1 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers.

18.2 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers
    - EDC (72,64) code protects against single and double bit errors
- Verification of feedback information for each data phase during crossbar transfer
- Error injection for testing
    - Programmable master and slave port specifiers
    - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
    - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable integrity feedback check enable on a per-master-port basis

18.3 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown in the following diagram.

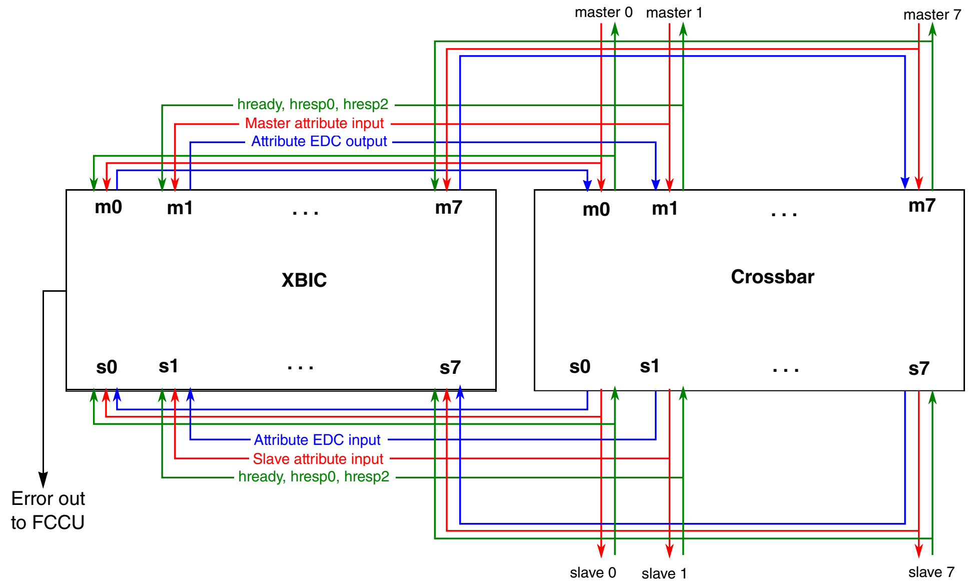


Figure 18-1. XBIC system block diagram

18.4 External signal description

The XBIC has no external interface signals.

18.5 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register), or in user mode generate an error termination.

XBIC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XBIC Module Control Register (XBIC_MCR)	32	R/W	FFFF_0000h	18.5.1/527
4	XBIC Error Injection Register (XBIC_EIR)	32	R/W	0000_0000h	18.5.2/529
8	XBIC Error Status Register (XBIC_ESR)	32	R	0000_0000h	18.5.3/529
C	XBIC Error Address Register (XBIC_EAR)	32	R	0000_0000h	18.5.4/532

18.5.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SE0	SE1	SE2	SE3	SE4	SE5	SE6	SE7	ME0	ME1	ME2	ME3	ME4	ME5	ME6	ME7
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_MCR field descriptions

Field	Description
0 SE0	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 0. 1 Attribute integrity checking enabled for slave port 0.
1 SE1	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 1. 1 Attribute integrity checking enabled for slave port 1.
2 SE2	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 2. 1 Attribute integrity checking enabled for slave port 2.
3 SE3	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 3. 1 Attribute integrity checking enabled for slave port 3.
4 SE4	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 4. 1 Attribute integrity checking enabled for slave port 4.
5 SE5	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 5. 1 Attribute integrity checking enabled for slave port 5.
6 SE6	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 6. 1 Attribute integrity checking enabled for slave port 6.
7 SE7	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 7. 1 Attribute integrity checking enabled for slave port 7.

Table continues on the next page...

XBIC_MCR field descriptions (continued)

Field	Description
8 ME0	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 0. 1 Attribute integrity checking enabled for master port 0.
9 ME1	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 1. 1 Attribute integrity checking enabled for master port 1.
10 ME2	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 2. 1 Attribute integrity checking enabled for master port 2.
11 ME3	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 3. 1 Attribute integrity checking enabled for master port 3.
12 ME4	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 4. 1 Attribute integrity checking enabled for master port 4.
13 ME5	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 5. 1 Attribute integrity checking enabled for master port 5.
14 ME6	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 6. 1 Attribute integrity checking enabled for master port 6.
15 ME7	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 7. 1 Attribute integrity checking enabled for master port 7.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.5.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIE								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		SLV			MST						SYN				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_EIR field descriptions

Field	Description
0 EIE	Error Injection Enable. 0 Error injection disabled 1 Error injection enabled
1–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 SLV	Target Slave Port. Error injection is enabled for the designated slave port. Other slave ports are unaffected.
20–23 MST	Target Master ID. Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
24–31 SYN	Syndrome. This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

18.5.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the last transfer. If an attribute integrity check error was detected, the slave port, the master port, and the syndrome are captured in the SLV, MST, and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are catured in the DPSE0-7 and DPME0-7 fields.

This register is cleared only on reset.

NOTE

When the XBIC detects back-to-back faults on a system bus path through the crossbar switch, the fault information captured in the XBIC_ESR does not correspond to the initial fault event but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR register describing the initial fault event is lost. This can only occur in the event of a series of bus transactions targeting the same crossbar slave target where the series of bus transactions are not separated by idle or stall cycles.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	DPSE0	DPSE1	DPSE2	DPSE3	DPSE4	DPSE5	DPSE6	DPSE7	DPME0	DPME1	DPME2	DPME3	DPME4	DPME5	DPME6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DPME7	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_ESR field descriptions

Field	Description
0 VLD	Error Status Valid. 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
1 DPSE0	Data phase slave port error.

Table continues on the next page...

XBIC_ESR field descriptions (continued)

Field	Description
	0 No error on slave port 0 1 Data phase error on slave port 0
2 DPSE1	Data phase slave port error. 0 No error on slave port 1 1 Data phase error on slave port 1
3 DPSE2	Data phase slave port error. 0 No error on slave port 2 1 Data phase error on slave port 2
4 DPSE3	Data phase slave port error. 0 No error on slave port 3 1 Data phase error on slave port 3
5 DPSE4	Data phase slave port error. 0 No error on slave port 4 1 Data phase error on slave port 4
6 DPSE5	Data phase slave port error. 0 No error on slave port 5 1 Data phase error on slave port 5
7 DPSE6	Data phase slave port error. 0 No error on slave port 6 1 Data phase error on slave port 6
8 DPSE7	Data phase slave port error. 0 No error on slave port 7 1 Data phase error on slave port 7
9 DPME0	Data phase master port error. 0 No error on master port 0 1 Data phase error on master port 0
10 DPME1	Data phase master port error. 0 No error on master port 1 1 Data phase error on master port 1
11 DPME2	Data phase master port error. 0 No error on master port 2 1 Data phase error on master port 2
12 DPME3	Data phase master port error. 0 No error on master port 3 1 Data phase error on master port 3
13 DPME4	Data phase master port error.

Table continues on the next page...

XBIC_ESR field descriptions (continued)

Field	Description
	0 No error on master port 4 1 Data phase error on master port 4
14 DPME5	Data phase master port error. 0 No error on master port 5 1 Data phase error on master port 5
15 DPME6	Data phase master port error. 0 No error on master port 6 1 Data phase error on master port 6
16 DPME7	Data phase master port error. 0 No error on master port 7 1 Data phase error on master port 7
17–19 SLV	Slave Port. The slave port targeted by the last transfer with an error.
20–23 MST	Master ID. The master ID value of the last transfer with an error.
24–31 SYN	Syndrome. The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see Table 18-1 .

18.5.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the first transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

NOTE

When the XBIC detects back-to-back faults on a system bus path through the crossbar switch, the fault information captured in the XBIC_EAR does not correspond to the initial fault event but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR register describing the initial fault event is lost. This can only occur in the event of a series of bus transactions targeting the same crossbar slave target where the series of bus transactions are not separated by idle or stall cycles.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_EAR field descriptions

Field	Description
0–31 ADDR	The address of the last transfer with an error.

18.6 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar interface. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a individual port basis. When an error is detected, it is captured in XBIC_ESR register and XBIC also sends out an error signal to the Fault Collection and Control Unit (FCCU). Detection of an error does not generate a bus error. The XBIC integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

During the AHB address phase, the crossbar attribute information is checked using an 8-bit Error Detection Code (EDC), which detects any single- or double-bit errors. When an error is detected, due either to hardware fault or error injection, information related to the error is captured and reported in XBIC_ESR and XBIC_EAR. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in Table 18-1. Any other syndrome indicates a multi-bit error.

During the AHB data phase, the responding signals from the slave to master are checked when passing through the crossbar. When an error is detected, the information is captured in XBIC_ESR DSME0-7 field.

The XBIC can also be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. This approach allows the check logic to be verified without compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared

Table 18-1. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54

Table continues on the next page...

Table 18-1. Hexadecimal attribute single-bit error syndromes (continued)

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

Chapter 19
Peripheral Bridge (AIPS-Lite)

19.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The peripheral bridge converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

The peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 16 KB. (It might be possible that all the peripheral slots are not used. See the memory map chapter for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

19.1.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width
- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Programming model provides memory protection functionality

19.1.2 General operation

The slave devices connected to the peripheral bridge are modules which contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge.

The register maps of the peripherals are located on 16 KB boundaries. Each peripheral is allocated one or more 16-KB block(s) of the memory map.

19.2 Memory map/register definition

The 32-bit peripheral bridge registers can be accessed only in supervisor mode by trusted bus masters. Additionally, these registers must be read from or written to only by a 32-bit aligned access. The peripheral bridge registers are mapped into the Peripheral Access Control Register A PACRA[PACR0] address space.

AIPS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Master Privilege Register A (AIPS_MPRA)	32	R/W	See section	19.2.1/538
100	Peripheral Access Control Register (AIPS_PACRA)	32	R/W	See section	19.2.2/540
104	Peripheral Access Control Register (AIPS_PACRB)	32	R/W	See section	19.2.2/540
108	Peripheral Access Control Register (AIPS_PACRC)	32	R/W	See section	19.2.2/540
10C	Peripheral Access Control Register (AIPS_PACRD)	32	R/W	See section	19.2.2/540
110	Peripheral Access Control Register (AIPS_PACR_Reserved)	32	R/W	See section	19.2.2/540
114	Peripheral Access Control Register (AIPS_PACRF)	32	R/W	See section	19.2.2/540
118	Peripheral Access Control Register (AIPS_PACRG)	32	R/W	See section	19.2.2/540
11C	Peripheral Access Control Register (AIPS_PACRH)	32	R/W	See section	19.2.2/540
140	Off-Platform Peripheral Access Control Register (AIPS_OPACRA)	32	R/W	See section	19.2.3/545
144	Off-Platform Peripheral Access Control Register (AIPS_OPACRB)	32	R/W	See section	19.2.3/545
148	Off-Platform Peripheral Access Control Register (AIPS_OPACRC)	32	R/W	See section	19.2.3/545
14C	Off-Platform Peripheral Access Control Register (AIPS_OPACRD)	32	R/W	See section	19.2.3/545
150	Off-Platform Peripheral Access Control Register (AIPS_OPACRE)	32	R/W	See section	19.2.3/545
154	Off-Platform Peripheral Access Control Register (AIPS_OPACRF)	32	R/W	See section	19.2.3/545
158	Off-Platform Peripheral Access Control Register (AIPS_OPACRG)	32	R/W	See section	19.2.3/545
15C	Off-Platform Peripheral Access Control Register (AIPS_OPACRH)	32	R/W	See section	19.2.3/545
160	Off-Platform Peripheral Access Control Register (AIPS_OPACRI)	32	R/W	See section	19.2.3/545
164	Off-Platform Peripheral Access Control Register (AIPS_OPACRJ)	32	R/W	See section	19.2.3/545

Table continues on the next page...

AIPS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
168	Off-Platform Peripheral Access Control Register (AIPS_OPACRK)	32	R/W	See section	19.2.3/545
16C	Off-Platform Peripheral Access Control Register (AIPS_OPACRL)	32	R/W	See section	19.2.3/545
170	Off-Platform Peripheral Access Control Register (AIPS_OPACRM)	32	R/W	See section	19.2.3/545
174	Off-Platform Peripheral Access Control Register (AIPS_OPACRN)	32	R/W	See section	19.2.3/545
178	Off-Platform Peripheral Access Control Register (AIPS_OPACRO)	32	R/W	See section	19.2.3/545
17C	Off-Platform Peripheral Access Control Register (AIPS_OPACRP)	32	R/W	See section	19.2.3/545
180	Off-Platform Peripheral Access Control Register (AIPS_OPACRQ)	32	R/W	See section	19.2.3/545
184	Off-Platform Peripheral Access Control Register (AIPS_OPACRR)	32	R/W	See section	19.2.3/545
188	Off-Platform Peripheral Access Control Register (AIPS_OPACRS)	32	R/W	See section	19.2.3/545
18C	Off-Platform Peripheral Access Control Register (AIPS_OPACRT)	32	R/W	See section	19.2.3/545
190	Off-Platform Peripheral Access Control Register (AIPS_OPACRU)	32	R/W	See section	19.2.3/545
194	Off-Platform Peripheral Access Control Register (AIPS_OPACRV)	32	R/W	See section	19.2.3/545
198	Off-Platform Peripheral Access Control Register (AIPS_OPACRW)	32	R/W	See section	19.2.3/545
19C	Off-Platform Peripheral Access Control Register (AIPS_OPACRX)	32	R/W	See section	19.2.3/545
1A0	Off-Platform Peripheral Access Control Register (AIPS_OPACRY)	32	R/W	See section	19.2.3/545
1A4	Off-Platform Peripheral Access Control Register (AIPS_OPACRZ)	32	R/W	See section	19.2.3/545
1A8	Off-Platform Peripheral Access Control Register (AIPS_OPACRAA)	32	R/W	See section	19.2.3/545
1AC	Off-Platform Peripheral Access Control Register (AIPS_OPACRAB)	32	R/W	See section	19.2.3/545
1B0	Off-Platform Peripheral Access Control Register (AIPS_OPACRAC)	32	R/W	See section	19.2.3/545
1B4	Off-Platform Peripheral Access Control Register (AIPS_OPACRAD)	32	R/W	See section	19.2.3/545
1B8	Off-Platform Peripheral Access Control Register (AIPS_OPACRAE)	32	R/W	See section	19.2.3/545
1BC	Off-Platform Peripheral Access Control Register (AIPS_OPACRAF)	32	R/W	See section	19.2.3/545

19.2.1 Master Privilege Register A (AIPS_MPRA)

This register specifies eight identical 4-bit fields defining the access-privilege level associated with a bus master in the device to various peripherals. The register provides one field per bus master. Each master is assigned a logical master number. See the device configuration information for details.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MTR0	MTW0	MPL0	Reserved				0	MTR2	MTW2	MPL2	0	MTR3	MTW3	MPL3
W																
Reset	0*	1*	1*	1*	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MTR4	MTW4	MPL4	0	MTR5	MTW5	MPL5	Reserved				Reserved			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0	0	0	0	0	0

* Notes:

- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_MPRA field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MTR0	Master 0 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
2 MTW0	Master 0 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL0	Master 0 Privilege Level Specifies how the privilege level of the master is determined.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
4–7 Reserved	This field is reserved.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 MTR2	Master 2 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
10 MTW2	Master 2 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
11 MPL2	Master 2 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 MTR3	Master 3 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
14 MTW3	Master 3 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
15 MPL3	Master 3 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 MTR4	Master 4 Trusted For Read Determines whether the master is trusted for read accesses.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
18 MTW4	Master 4 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
19 MPL4	Master 4 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MTR5	Master 5 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
22 MTW5	Master 5 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
23 MPL5	Master 5 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
24–27 Reserved	This field is reserved.
28–31 Reserved	This field is reserved.

19.2.2 Peripheral Access Control Register (AIPS_PACR*n*)

Each of the on-platform peripherals has a 4-bit PACR *n* field, which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACR *x* register.

The peripheral assignments to each PACR are defined by the memory map slot that the peripherals are assigned. See the device's memory map details for the assignments for a particular device. If a peripheral is absent, the corresponding PACR field is not implemented. Reads to the location return zeroes and writes are ignored.

The following table shows the top-level structure of PACRs.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x10C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x11C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

Address: 0h base + 100h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W	▨				▨				▨				▨			
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W	▨				▨				▨				▨			
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

AIPS_PACR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accesssses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write accesssses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

19.2.3 Off-Platform Peripheral Access Control Register (AIPS_OPACR*n*)

Each of the peripherals has a 4-bit OPACR *n* field, which defines the access levels supported by this module. Eight OPACR fields are grouped together to form a 32-bit OPACR *x* register.

The peripheral assignments to each OPACR are defined by the memory map slot that the peripherals are assigned. See the device's memory map details for the assignments for a particular device. If a peripheral is absent, the corresponding OPACR field is not implemented. Reads to the location return zeroes and writes are ignored.

The following table shows the top-level structure of OPACRs.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x140	OPACRA	OPACR0	OPACR1	OPACR2	OPACR3	OPACR4	OPACR5	OPACR6	OPACR7
0x144	OPACRB	OPACR8	OPACR9	OPACR10	OPACR11	OPACR12	OPACR13	OPACR14	OPACR15
0x148	OPACRC	OPACR16	OPACR17	OPACR18	OPACR19	OPACR20	OPACR21	OPACR22	OPACR23
0x14C	OPACRD	OPACR24	OPACR25	OPACR26	OPACR27	OPACR28	OPACR29	OPACR30	OPACR31
0x150	OPACRE	OPACR32	OPACR33	OPACR34	OPACR35	OPACR36	OPACR37	OPACR38	OPACR39
0x154	OPACRF	OPACR40	OPACR41	OPACR42	OPACR43	OPACR44	OPACR45	OPACR46	OPACR47
0x158	OPACRG	OPACR48	OPACR49	OPACR50	OPACR51	OPACR52	OPACR53	OPACR54	OPACR55
0x15C	OPACRH	OPACR56	OPACR57	OPACR58	OPACR59	OPACR60	OPACR61	OPACR62	OPACR63
0x160	OPACRI	OPACR64	OPACR65	OPACR66	OPACR67	OPACR68	OPACR69	OPACR70	OPACR71
0x164	OPACRJ	OPACR72	OPACR73	OPACR74	OPACR75	OPACR76	OPACR77	OPACR78	OPACR79
0x168	OPACRK	OPACR80	OPACR81	OPACR82	OPACR83	OPACR84	OPACR85	OPACR86	OPACR87
0x16C	OPACRL	OPACR88	OPACR89	OPACR90	OPACR91	OPACR92	OPACR93	OPACR94	OPACR95
0x170	OPACRM	OPACR96	OPACR97	OPACR98	OPACR99	OPACR100	OPACR101	OPACR102	OPACR103
0x174	OPACRN	OPACR104	OPACR105	OPACR106	OPACR107	OPACR108	OPACR109	OPACR110	OPACR111
0x178	OPACRO	OPACR112	OPACR113	OPACR114	OPACR115	OPACR116	OPACR117	OPACR118	OPACR119
0x17C	OPACRP	OPACR120	OPACR121	OPACR122	OPACR123	OPACR124	OPACR125	OPACR126	OPACR127
0x180	OPACRQ	OPACR128	OPACR129	OPACR130	OPACR131	OPACR132	OPACR133	OPACR134	OPACR135
0x184	OPACRR	OPACR136	OPACR137	OPACR138	OPACR139	OPACR140	OPACR141	OPACR142	OPACR143
0x188	OPACRS	OPACR144	OPACR145	OPACR146	OPACR147	OPACR148	OPACR149	OPACR150	OPACR151
0x18C	OPACRT	OPACR152	OPACR153	OPACR154	OPACR155	OPACR156	OPACR157	OPACR158	OPACR159
0x190	OPACRU	OPACR160	OPACR161	OPACR162	OPACR163	OPACR164	OPACR165	OPACR166	OPACR167
0x194	OPACRV	OPACR168	OPACR169	OPACR170	OPACR171	OPACR172	OPACR173	OPACR174	OPACR175
0x198	OPACRW	OPACR176	OPACR177	OPACR178	OPACR179	OPACR180	OPACR181	OPACR182	OPACR183
0x19C	OPACRX	OPACR184	OPACR185	OPACR186	OPACR187	OPACR188	OPACR189	OPACR190	OPACR191
0x1A0	OPACRY	OPACR192	OPACR193	OPACR194	OPACR195	OPACR196	OPACR197	OPACR198	OPACR199
0x1A4	OPACRZ	OPACR200	OPACR201	OPACR202	OPACR203	OPACR204	OPACR205	OPACR206	OPACR207
0x1A8	OPACRAA	OPACR208	OPACR209	OPACR210	OPACR211	OPACR212	OPACR213	OPACR214	OPACR215
0x1AC	OPACRAB	OPACR216	OPACR217	OPACR218	OPACR219	OPACR220	OPACR221	OPACR222	OPACR223
0x1B0	OPACRAC	OPACR224	OPACR225	OPACR226	OPACR227	OPACR228	OPACR229	OPACR230	OPACR231

Table continues on the next page...

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x1B4	OPACRAD	OPACR232	OPACR233	OPACR234	OPACR235	OPACR236	OPACR237	OPACR238	OPACR239
0x1B8	OPACRAE	OPACR240	OPACR241	OPACR242	OPACR243	OPACR244	OPACR245	OPACR246	OPACR247
0x1BC	OPACRAF	OPACR248	OPACR249	OPACR250	OPACR251	OPACR252	OPACR253	OPACR254	OPACR255

Address: 0h base + 140h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W	▨				▨				▨				▨			
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W	▨				▨				▨				▨			
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

* Notes:

- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_OPACR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attributeMPR *x* [MPL *n*], and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

19.3 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

19.3.1 Access support

Aligned and misaligned 32-bit, 16-bit, and byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. The peripheral bridge performs two slave peripheral bus transfers for allowed 64-bit transactions, to 32-bit sized peripheral slots only.All other accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted that is larger than the targeted port, an error response is generated.

Chapter 20
System Memory Protection Unit (SMPU)

20.1 Overview

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors system bus transactions and evaluates their appropriateness using preprogrammed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

20.1.1 Block diagram

A simplified block diagram of the SMPU module is shown in the following figure. The access signals coming from the crossbar switch slave ports are shown on the left. The access evaluation macros and region descriptors are in the middle, and the peripheral bus interface is on the right side.

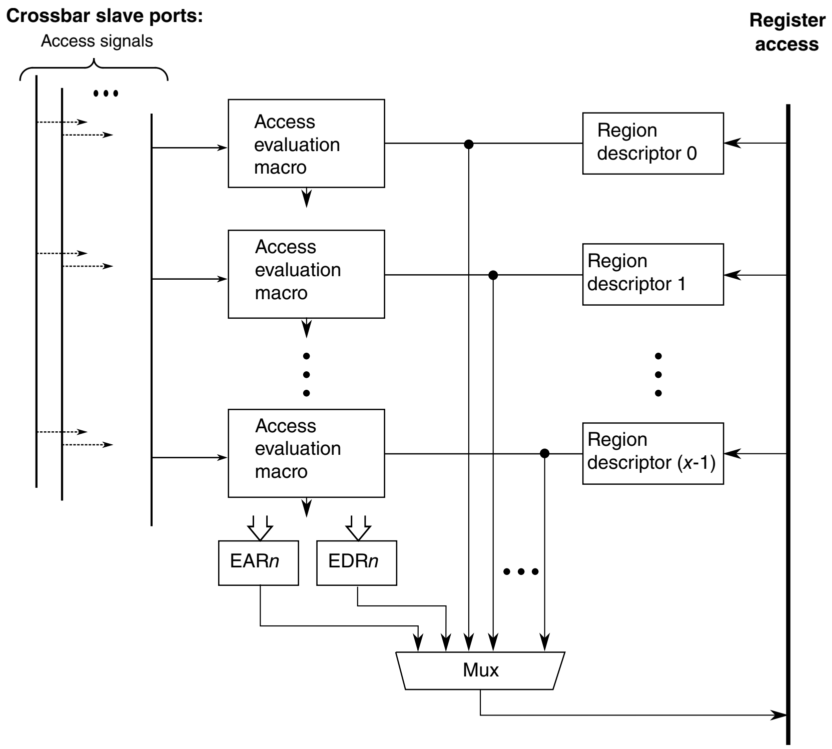


Figure 20-1. SMPU block diagram

The access evaluation macros determine hits to memory regions and detect violations to access protections. If a violation is detected, the macros update the appropriate master's error registers. For details of the access evaluation macro, see Access evaluation macro.

20.1.2 Features

The SMPU feature set includes:

- Arbitrarily sized protection regions alignable anywhere in memory.
  - Region sizes can vary from a minimum of 1 byte to a maximum of 4 GB.
- Read/write access control permissions are defined in the region descriptor.
- Global cache-inhibit attribute indicator.
- Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues.
- Priority given to granting permission over denying access for overlapping region descriptors.

- Detection of access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Instruction storage (ISI) or data storage (DSI) interrupts generated by aborted core accesses. Buffered writes generate Machine check exception (refer to the documentation related to the core for details).
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enable/disable control bit.

20.2 Memory map/register definition

The programming model is partitioned into three groups:

- Control registers
- Error capture registers
- Data structure containing the region descriptors

The programming model can be referenced only in supervisor mode using 32-bit accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register, or a read of a write-only register), or in user mode generate an error termination.

NOTE

See the chip configuration details for any chip-specific register information for this module. For example, a specific instance of the module may support only eight or fewer region descriptors.

SMPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control/Error Status Register 0 (SMPU_CESR0)	32	R/W	0000_8002h	20.2.1/558
4	Control/Error Status Register 1 (SMPU_CESR1)	32	R/W	0000_8004h	20.2.2/559
100	Error Address Register, Bus Master n (SMPU_EAR0)	32	R	0000_0000h	20.2.3/560
104	Error Detail Register, Bus Master n (SMPU_EDR0)	32	R	0000_0080h	20.2.4/561
108	Error Address Register, Bus Master n (SMPU_EAR1)	32	R	0000_0000h	20.2.3/560
10C	Error Detail Register, Bus Master n (SMPU_EDR1)	32	R	0000_0080h	20.2.4/561
110	Error Address Register, Bus Master n (SMPU_EAR2)	32	R	0000_0000h	20.2.3/560
114	Error Detail Register, Bus Master n (SMPU_EDR2)	32	R	0000_0080h	20.2.4/561

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
118	Error Address Register, Bus Master n (SMPU_EAR3)	32	R	0000_0000h	20.2.3/560
11C	Error Detail Register, Bus Master n (SMPU_EDR3)	32	R	0000_0080h	20.2.4/561
120	Error Address Register, Bus Master n (SMPU_EAR4)	32	R	0000_0000h	20.2.3/560
124	Error Detail Register, Bus Master n (SMPU_EDR4)	32	R	0000_0080h	20.2.4/561
128	Error Address Register, Bus Master n (SMPU_EAR5)	32	R	0000_0000h	20.2.3/560
12C	Error Detail Register, Bus Master n (SMPU_EDR5)	32	R	0000_0080h	20.2.4/561
130	Error Address Register, Bus Master n (SMPU_EAR6)	32	R	0000_0000h	20.2.3/560
134	Error Detail Register, Bus Master n (SMPU_EDR6)	32	R	0000_0080h	20.2.4/561
138	Error Address Register, Bus Master n (SMPU_EAR7)	32	R	0000_0000h	20.2.3/560
13C	Error Detail Register, Bus Master n (SMPU_EDR7)	32	R	0000_0080h	20.2.4/561
140	Error Address Register, Bus Master n (SMPU_EAR8)	32	R	0000_0000h	20.2.3/560
144	Error Detail Register, Bus Master n (SMPU_EDR8)	32	R	0000_0080h	20.2.4/561
148	Error Address Register, Bus Master n (SMPU_EAR9)	32	R	0000_0000h	20.2.3/560
14C	Error Detail Register, Bus Master n (SMPU_EDR9)	32	R	0000_0080h	20.2.4/561
150	Error Address Register, Bus Master n (SMPU_EAR10)	32	R	0000_0000h	20.2.3/560
154	Error Detail Register, Bus Master n (SMPU_EDR10)	32	R	0000_0080h	20.2.4/561
158	Error Address Register, Bus Master n (SMPU_EAR11)	32	R	0000_0000h	20.2.3/560
15C	Error Detail Register, Bus Master n (SMPU_EDR11)	32	R	0000_0080h	20.2.4/561
160	Error Address Register, Bus Master n (SMPU_EAR12)	32	R	0000_0000h	20.2.3/560
164	Error Detail Register, Bus Master n (SMPU_EDR12)	32	R	0000_0080h	20.2.4/561
168	Error Address Register, Bus Master n (SMPU_EAR13)	32	R	0000_0000h	20.2.3/560
16C	Error Detail Register, Bus Master n (SMPU_EDR13)	32	R	0000_0080h	20.2.4/561
170	Error Address Register, Bus Master n (SMPU_EAR14)	32	R	0000_0000h	20.2.3/560
174	Error Detail Register, Bus Master n (SMPU_EDR14)	32	R	0000_0080h	20.2.4/561
178	Error Address Register, Bus Master n (SMPU_EAR15)	32	R	0000_0000h	20.2.3/560
17C	Error Detail Register, Bus Master n (SMPU_EDR15)	32	R	0000_0080h	20.2.4/561
400	Region Descriptor n, Word 0 (SMPU_RGD0_WORD0)	32	R/W	0000_0000h	20.2.5/562
404	Region Descriptor n, Word 1 (SMPU_RGD0_WORD1)	32	R/W	0000_0000h	20.2.6/562
408	Region Descriptor n, Word 2 (SMPU_RGD0_WORD2)	32	R/W	0000_0000h	20.2.7/563
40C	Region Descriptor n, Word 3 (SMPU_RGD0_WORD3)	32	R/W	0000_0000h	20.2.8/565
410	Region Descriptor n, Word 0 (SMPU_RGD1_WORD0)	32	R/W	0000_0000h	20.2.5/562
414	Region Descriptor n, Word 1 (SMPU_RGD1_WORD1)	32	R/W	0000_0000h	20.2.6/562
418	Region Descriptor n, Word 2 (SMPU_RGD1_WORD2)	32	R/W	0000_0000h	20.2.7/563
41C	Region Descriptor n, Word 3 (SMPU_RGD1_WORD3)	32	R/W	0000_0000h	20.2.8/565
420	Region Descriptor n, Word 0 (SMPU_RGD2_WORD0)	32	R/W	0000_0000h	20.2.5/562
424	Region Descriptor n, Word 1 (SMPU_RGD2_WORD1)	32	R/W	0000_0000h	20.2.6/562
428	Region Descriptor n, Word 2 (SMPU_RGD2_WORD2)	32	R/W	0000_0000h	20.2.7/563
42C	Region Descriptor n, Word 3 (SMPU_RGD2_WORD3)	32	R/W	0000_0000h	20.2.8/565
430	Region Descriptor n, Word 0 (SMPU_RGD3_WORD0)	32	R/W	0000_0000h	20.2.5/562

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
434	Region Descriptor n, Word 1 (SMPU_RGD3_WORD1)	32	R/W	0000_0000h	20.2.6/562
438	Region Descriptor n, Word 2 (SMPU_RGD3_WORD2)	32	R/W	0000_0000h	20.2.7/563
43C	Region Descriptor n, Word 3 (SMPU_RGD3_WORD3)	32	R/W	0000_0000h	20.2.8/565
440	Region Descriptor n, Word 0 (SMPU_RGD4_WORD0)	32	R/W	0000_0000h	20.2.5/562
444	Region Descriptor n, Word 1 (SMPU_RGD4_WORD1)	32	R/W	0000_0000h	20.2.6/562
448	Region Descriptor n, Word 2 (SMPU_RGD4_WORD2)	32	R/W	0000_0000h	20.2.7/563
44C	Region Descriptor n, Word 3 (SMPU_RGD4_WORD3)	32	R/W	0000_0000h	20.2.8/565
450	Region Descriptor n, Word 0 (SMPU_RGD5_WORD0)	32	R/W	0000_0000h	20.2.5/562
454	Region Descriptor n, Word 1 (SMPU_RGD5_WORD1)	32	R/W	0000_0000h	20.2.6/562
458	Region Descriptor n, Word 2 (SMPU_RGD5_WORD2)	32	R/W	0000_0000h	20.2.7/563
45C	Region Descriptor n, Word 3 (SMPU_RGD5_WORD3)	32	R/W	0000_0000h	20.2.8/565
460	Region Descriptor n, Word 0 (SMPU_RGD6_WORD0)	32	R/W	0000_0000h	20.2.5/562
464	Region Descriptor n, Word 1 (SMPU_RGD6_WORD1)	32	R/W	0000_0000h	20.2.6/562
468	Region Descriptor n, Word 2 (SMPU_RGD6_WORD2)	32	R/W	0000_0000h	20.2.7/563
46C	Region Descriptor n, Word 3 (SMPU_RGD6_WORD3)	32	R/W	0000_0000h	20.2.8/565
470	Region Descriptor n, Word 0 (SMPU_RGD7_WORD0)	32	R/W	0000_0000h	20.2.5/562
474	Region Descriptor n, Word 1 (SMPU_RGD7_WORD1)	32	R/W	0000_0000h	20.2.6/562
478	Region Descriptor n, Word 2 (SMPU_RGD7_WORD2)	32	R/W	0000_0000h	20.2.7/563
47C	Region Descriptor n, Word 3 (SMPU_RGD7_WORD3)	32	R/W	0000_0000h	20.2.8/565
480	Region Descriptor n, Word 0 (SMPU_RGD8_WORD0)	32	R/W	0000_0000h	20.2.5/562
484	Region Descriptor n, Word 1 (SMPU_RGD8_WORD1)	32	R/W	0000_0000h	20.2.6/562
488	Region Descriptor n, Word 2 (SMPU_RGD8_WORD2)	32	R/W	0000_0000h	20.2.7/563
48C	Region Descriptor n, Word 3 (SMPU_RGD8_WORD3)	32	R/W	0000_0000h	20.2.8/565
490	Region Descriptor n, Word 0 (SMPU_RGD9_WORD0)	32	R/W	0000_0000h	20.2.5/562
494	Region Descriptor n, Word 1 (SMPU_RGD9_WORD1)	32	R/W	0000_0000h	20.2.6/562
498	Region Descriptor n, Word 2 (SMPU_RGD9_WORD2)	32	R/W	0000_0000h	20.2.7/563
49C	Region Descriptor n, Word 3 (SMPU_RGD9_WORD3)	32	R/W	0000_0000h	20.2.8/565
4A0	Region Descriptor n, Word 0 (SMPU_RGD10_WORD0)	32	R/W	0000_0000h	20.2.5/562
4A4	Region Descriptor n, Word 1 (SMPU_RGD10_WORD1)	32	R/W	0000_0000h	20.2.6/562
4A8	Region Descriptor n, Word 2 (SMPU_RGD10_WORD2)	32	R/W	0000_0000h	20.2.7/563
4AC	Region Descriptor n, Word 3 (SMPU_RGD10_WORD3)	32	R/W	0000_0000h	20.2.8/565
4B0	Region Descriptor n, Word 0 (SMPU_RGD11_WORD0)	32	R/W	0000_0000h	20.2.5/562
4B4	Region Descriptor n, Word 1 (SMPU_RGD11_WORD1)	32	R/W	0000_0000h	20.2.6/562
4B8	Region Descriptor n, Word 2 (SMPU_RGD11_WORD2)	32	R/W	0000_0000h	20.2.7/563
4BC	Region Descriptor n, Word 3 (SMPU_RGD11_WORD3)	32	R/W	0000_0000h	20.2.8/565
4C0	Region Descriptor n, Word 0 (SMPU_RGD12_WORD0)	32	R/W	0000_0000h	20.2.5/562
4C4	Region Descriptor n, Word 1 (SMPU_RGD12_WORD1)	32	R/W	0000_0000h	20.2.6/562
4C8	Region Descriptor n, Word 2 (SMPU_RGD12_WORD2)	32	R/W	0000_0000h	20.2.7/563
4CC	Region Descriptor n, Word 3 (SMPU_RGD12_WORD3)	32	R/W	0000_0000h	20.2.8/565

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4D0	Region Descriptor n, Word 0 (SMPU_RGD13_WORD0)	32	R/W	0000_0000h	20.2.5/562
4D4	Region Descriptor n, Word 1 (SMPU_RGD13_WORD1)	32	R/W	0000_0000h	20.2.6/562
4D8	Region Descriptor n, Word 2 (SMPU_RGD13_WORD2)	32	R/W	0000_0000h	20.2.7/563
4DC	Region Descriptor n, Word 3 (SMPU_RGD13_WORD3)	32	R/W	0000_0000h	20.2.8/565
4E0	Region Descriptor n, Word 0 (SMPU_RGD14_WORD0)	32	R/W	0000_0000h	20.2.5/562
4E4	Region Descriptor n, Word 1 (SMPU_RGD14_WORD1)	32	R/W	0000_0000h	20.2.6/562
4E8	Region Descriptor n, Word 2 (SMPU_RGD14_WORD2)	32	R/W	0000_0000h	20.2.7/563
4EC	Region Descriptor n, Word 3 (SMPU_RGD14_WORD3)	32	R/W	0000_0000h	20.2.8/565
4F0	Region Descriptor n, Word 0 (SMPU_RGD15_WORD0)	32	R/W	0000_0000h	20.2.5/562
4F4	Region Descriptor n, Word 1 (SMPU_RGD15_WORD1)	32	R/W	0000_0000h	20.2.6/562
4F8	Region Descriptor n, Word 2 (SMPU_RGD15_WORD2)	32	R/W	0000_0000h	20.2.7/563
4FC	Region Descriptor n, Word 3 (SMPU_RGD15_WORD3)	32	R/W	0000_0000h	20.2.8/565

20.2.1 Control/Error Status Register 0 (SMPU_CESR0)

Address: 0h base + 0h offset = 0h

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>R</td><td colspan="16">MERR</td></tr>
<tr><td>W</td><td colspan="16">w1c</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

<table>
<tr><td>Bit</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>R</td><td>1</td><td colspan="11">0</td><td colspan="3">HRL</td><td rowspan="2">GVLD</td></tr>
<tr><td>W</td><td></td><td colspan="11"></td><td colspan="3"></td></tr>
<tr><td>Reset</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
</table>

SMPU_CESR0 field descriptions

Field	Description
0–15 MERR	Master n error, where the bus master number matches the bit number Indicates a captured error in EARn and EDRn. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing one to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master n. 1 An error has occurred for bus master n.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled.

20.2.2 Control/Error Status Register 1 (SMPU_CESR1)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEOVR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0											NRGD			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

SMPU_CESR1 field descriptions

Field	Description
0–15 MEOVR	Master n error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master n has occurred before the previous error was processed. The details of the first error are recorded in the EARn and EDRn registers and no information on subsequent errors is recorded until the associated CESR0[MERR] flag is cleared. 0 No error overrun condition has been detected for bus master n. 1 An error overrun condition has been detected for bus master n.

Table continues on the next page...

SMPU_CESR1 field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. The number of region descriptors is (4 x NRGD). 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

20.2.3 Error Address Register, Bus Master n (SMPU_EAR*n*)

When the SMPU detects an access error on bus master n, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0[MERR] set. Additional information about the faulting access is captured in the corresponding EDRn at the same time.[1]

Address: 0h base + 100h offset + (8d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_EAR*n* field descriptions

Field	Description
0–31 EADDR	Error address Indicates the reference address from bus master n that generated the access error.

[1] The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the CESR0[MERR] flag associated with the original error is cleared (by writing a 1).

20.2.4 Error Detail Register, Bus Master n (SMPU_EDR*n*)

When the SMPU detects an access error associated with bus master n, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0[MERR] is set. Information on the faulting address is captured in the corresponding EARn register at the same time. [2]

Address: 0h base + 104h offset + (8d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EACD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EACD								1	EATTR		ERW	EMN			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

SMPU_EDR*n* field descriptions

Field	Description
0–23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDRn contains a captured error and EACD is all zeroes, the access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
25–26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference.

Table continues on the next page...

[2] The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the CESR0[MERR] flag associated with the original error is cleared (by writing a 1).

SMPU_EDR*n* field descriptions (continued)

Field	Description
	0 Read 1 Write
28–31 EMN	Error master number Indicates the logical bus master number of the faulting reference.

20.2.5 Region Descriptor n, Word 0 (SMPU_RGD*n*_WORD0)

The first word of the region descriptor defines the byte start address of the memory region. Writes to this register clear the region descriptor's valid bit (RGDn_WORD3[VLD]).

Address: 0h base + 400h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD0 field descriptions

Field	Description
0–31 SRTADDR	Start address Defines the byte start address of the memory region.

20.2.6 Region Descriptor n, Word 1 (SMPU_RGD*n*_WORD1)

The second word of the region descriptor defines the end address of the memory region. (RGDn_WORD3[VLD]).

Address: 0h base + 404h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ENDADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD1 field descriptions

Field	Description
0–31 ENDADDR	End address Defines the byte end address of the memory region. **NOTE:** The SMPU does not verify that ENDADDR ≥ SRTADDR.

20.2.7 Region Descriptor n, Word 2 (SMPU_RGD*n*_WORD2)

RGD_WORD2 defines the access control rights of the memory region. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) operation.

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor’s valid bit.

Address: 0h base + 408h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD2 field descriptions

Field	Description
0–1 M0P	Bus master 0 permissions

Table continues on the next page...

SMPU_RGD*n*_WORD2 field descriptions (continued)

Field	Description
2–3 M1P	Bus master 1 permissions
4–5 M2P	Bus master 2 permissions
6–7 M3P	Bus master 3 permissions
8–9 M4P	Bus master 4 permissions
10–11 M5P	Bus master 5 permissions
12–13 M6P	Bus master 6 permissions
14–15 M7P	Bus master 7 permissions
16–17 M8P	Bus master 8 permissions
18–19 M9P	Bus master 9 permissions
20–21 M10P	Bus master 10 permissions
22–23 M11P	Bus master 11 permissions
24–25 M12P	Bus master 12 permissions
26–27 M13P	Bus master 13 permissions
28–29 M14P	Bus master 14 permissions
30–31 M15P	Bus master 15 permissions

20.2.8 Region Descriptor n, Word 3 (SMPU_RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, plus other miscellaneous flags.

Address: 0h base + 40Ch offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RO	0	CI	VLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD3 field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **NOTE:** The valid bit of the RGD and the global valid bit have no effect on this bit functionality (once set, even if the RGD is not valid or SMPU is not global enabled, the RO functionality is present). 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **NOTE:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size factoring into the end address. 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals the region descriptor is valid. Any write to RGDn_WORD0–2 clears this bit. 0 Region descriptor is invalid. 1 Region descriptor is valid.

20.3 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

20.3.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the connection matrix. The following block diagram represents an individual macro.

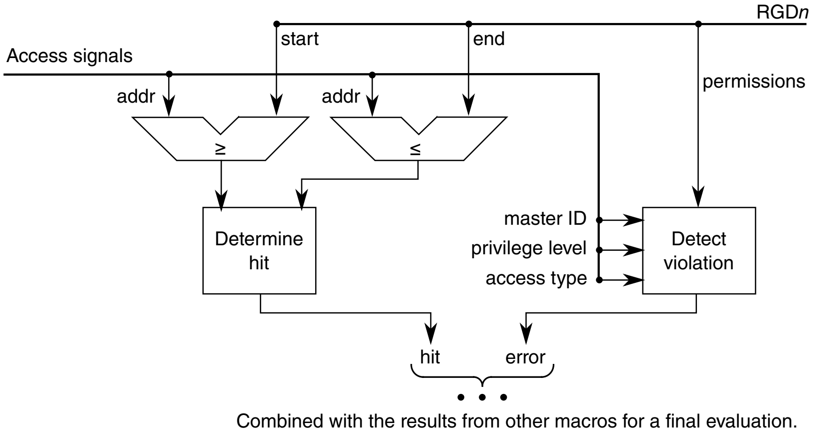


Figure 20-2. SMPU access evaluation macro

Each macro uses the information contained in the access signals (the address, the access type, and the master's ID and privilege level) and the contents of its corresponding region descriptor (RGD*n*) to perform two major functions: Hit determination and Violation determination. The individual hit and error results from each macro are then combined for a final evaluation; see Final evaluation and error terminations.

20.3.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. When RGD*n*_Word3[VLD] is 1, a region hit occurs when the requested address is between the start address (RGD*n*_Word0[SRTADDR]) and the end address (RGD*n*_Word1[ENDADDR]), inclusive.

NOTE

If ENDADDR < SRTADDR, no hit occurs.

NOTE

Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

20.3.1.2 Violation determination

While the access evaluation macro is determining region hit, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. Using the master ID, a set of effective permissions is generated from the appropriate fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions using the specification shown below.

Table 20-1. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no execute permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

20.3.2 Final evaluation and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.

3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see Application information.

NOTE

Similarly, the cache inhibit function gives a higher priority to the less restrictive result of overlapping regions. If two regions are hit and one has cache inhibit set, and the other does not, the result will not include a cache inhibit. It should also be noted that only the hit logic is taken into account for cache inhibit. For example, two region hits, one has protection violation and no cache inhibit, one has no violation and cache inhibit set. The result will be a granted access with no cache inhibit.

20.3.3 Power management

Disabling the SMPU by clearing CESR0[GVLD] minimizes power dissipation. To minimize the power dissipation of an enabled SMPU, invalidate unused region descriptors by clearing the associated RGDn_Word3[VLD] bits.

20.4 Initialization information

At system startup, load the appropriate number of region descriptors, including setting RGD*n*_Word3[VLD]. Setting CESR0[GVLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0[GVLD]=0). Next, load the appropriate region descriptors (RGDn), including the setting of the RGDn_Word3[VLD] bits. Finally, set CESR0[GVLD] to enable the entire module.

Note

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

20.5 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities:

- Creating a new memory region

  Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Modifying a region descriptor

  Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word 3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Removing a region descriptor

  Clearing RGD*n*_Word3[VLD] deletes an existing region descriptor.

- Accessing the SMPU

  Allocate a region descriptor to restrict SMPU access to supervisor mode from specific masters as appropriate.

- Detecting an access error

  The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0[MERR] signals which error registers contain captured fault data. Once the error information is retrieved, write a 1 to the appropriate CESR0[MERR] bit to clear it, thereby rearming the error capture logic associated with the EAR*n* and EDR*n* registers.

- Overlapping region descriptors

Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

Chapter 21
Interrupt Controller (INTC)

21.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or more processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 32 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

21.2 Block diagram

The following figure shows the block diagram of an INTC with four processors (PRC0...PRC3). The actual number of processors is described in the chip-specific INTC information. The structure of the INTC remains the same for each implemented processor.

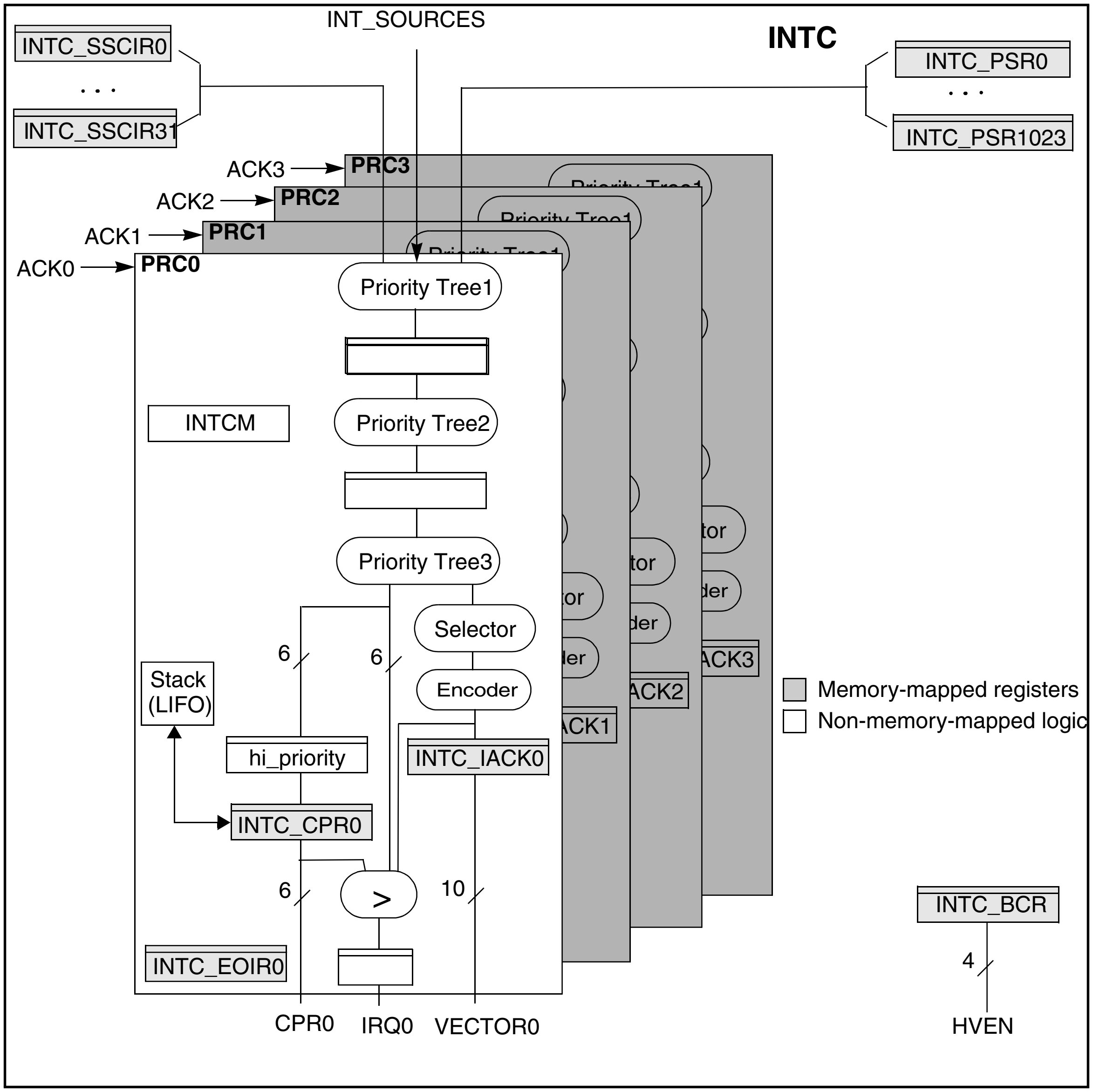


Figure 21-1. Block diagram for an INTC with four processors

21.3 Features

- Each peripheral interrupt source is software-steerable to any combination of interrupt request outputs to the following:
  - Processor 0
- 16 software-settable interrupt request sources
- 10-bit vector

    - Unique vector for each interrupt request source
    - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 32 priorities
- Each interrupt source can be triggered by software
- Preemption
    - Preemptive prioritized interrupt requests to processor
    - ISR with higher priority preempts ISRs or tasks with lower priorities
    - Automatic pushing or popping of preempted priority to or from a LIFO
    - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
    - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
    - 3 INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; four clock cycles from receipt of software request
- Monitor
    - The Interrupt Controller Monitor (INTCM) provides a way to set a maximum timer count for the interrupt latency from interrupt request (IRQ) to interrupt acknowledge (IACK) or from IRQ to return from interrupt (RFI). The purpose of this function is to provide a mechanism to monitor the latency of interrupt sources for a corresponding Processor to ensure that these critical interrupts execute within the expected window of time, increasing the reliability of the device. The hardware for this function is isolated in its own hierarchy to decrease the risk of fault interference.

21.4 Modes of operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ).

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

21.4.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN_PRCn bit in the INTC_BCR is negated. The hardware vector enable signal to any processor is driven as negated when its associated HVEN_PRC*n* bit is negated. The vector is read from the INTC_IACKRn registers. Reading the INTC_IACKR*n* negates the interrupt request to the associated processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in the INTC_CPRn onto the associated LIFO, and updates PRI in the associated INTC_CPR_PRC*n* with the new priority.

21.4.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for a given processor when its associated HVENn bit in the INTC_BCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted.

When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKRn, depending on which processor was assigned to handle a given interrupt source.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR*n* register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR*n* register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR*n* does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs is written at a time such that the PRI value in the associated INTC_CPR*n* register would need to be pushed and the previously last pushed

PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR*n* is updated with the new priority, and the associated LIFO is neither pushed or popped.

21.5 Memory map and register definition

A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

With the exception of the INTC_SSCIR *n* and INTC_PSR *n* registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSCIR*n* are 8 bits wide and INTC_PSR*n* are 16 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

When writing to INTC_PSRs or INTC_SSCIRs, the following restrictions (illustrated in Figure 21-2) apply:

- For PSRs, write accesses must not span 16-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).
- For SSCIRs, write accesses must not span 8-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).

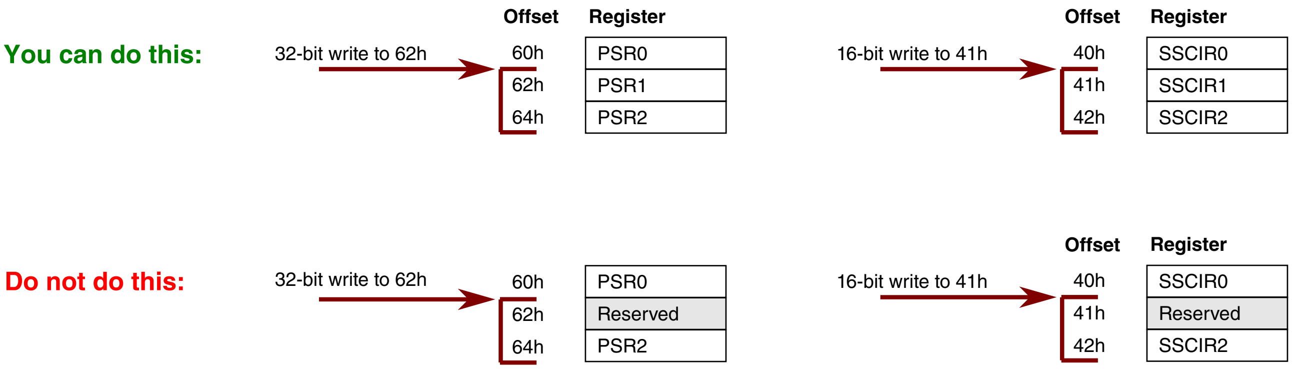


Figure 21-2. INTC_PSR*n* and INTC_SSCIR*n* write restrictions

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR *n*) are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR *n*) does not affect the operation of the write.

INTC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	INTC Block Configuration Register (INTC_BCR)	32	R/W	0000_0000h	21.5.1/604
10	INTC Current Priority Register for Processor 0 (INTC_CPR0)	32	R/W	0000_001Fh	21.5.2/605
20	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	32	R/W	0000_0000h	21.5.3/606
30	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	32	W	0000_0000h	21.5.4/607
40	INTC Software Set/Clear Interrupt Register (INTC_SSCIR0)	8	R/W	00h	21.5.5/607
41	INTC Software Set/Clear Interrupt Register (INTC_SSCIR1)	8	R/W	00h	21.5.5/607
42	INTC Software Set/Clear Interrupt Register (INTC_SSCIR2)	8	R/W	00h	21.5.5/607
43	INTC Software Set/Clear Interrupt Register (INTC_SSCIR3)	8	R/W	00h	21.5.5/607
44	INTC Software Set/Clear Interrupt Register (INTC_SSCIR4)	8	R/W	00h	21.5.5/607
45	INTC Software Set/Clear Interrupt Register (INTC_SSCIR5)	8	R/W	00h	21.5.5/607
46	INTC Software Set/Clear Interrupt Register (INTC_SSCIR6)	8	R/W	00h	21.5.5/607
47	INTC Software Set/Clear Interrupt Register (INTC_SSCIR7)	8	R/W	00h	21.5.5/607
48	INTC Software Set/Clear Interrupt Register (INTC_SSCIR8)	8	R/W	00h	21.5.5/607
49	INTC Software Set/Clear Interrupt Register (INTC_SSCIR9)	8	R/W	00h	21.5.5/607
4A	INTC Software Set/Clear Interrupt Register (INTC_SSCIR10)	8	R/W	00h	21.5.5/607
4B	INTC Software Set/Clear Interrupt Register (INTC_SSCIR11)	8	R/W	00h	21.5.5/607
4C	INTC Software Set/Clear Interrupt Register (INTC_SSCIR12)	8	R/W	00h	21.5.5/607
4D	INTC Software Set/Clear Interrupt Register (INTC_SSCIR13)	8	R/W	00h	21.5.5/607
4E	INTC Software Set/Clear Interrupt Register (INTC_SSCIR14)	8	R/W	00h	21.5.5/607
4F	INTC Software Set/Clear Interrupt Register (INTC_SSCIR15)	8	R/W	00h	21.5.5/607
60	INTC Priority Select Register (INTC_PSR0)	16	R/W	8000h	21.5.6/608
62	INTC Priority Select Register (INTC_PSR1)	16	R/W	8000h	21.5.6/608
64	INTC Priority Select Register (INTC_PSR2)	16	R/W	8000h	21.5.6/608
66	INTC Priority Select Register (INTC_PSR3)	16	R/W	8000h	21.5.6/608
68	INTC Priority Select Register (INTC_PSR4)	16	R/W	8000h	21.5.6/608
6A	INTC Priority Select Register (INTC_PSR5)	16	R/W	8000h	21.5.6/608
6C	INTC Priority Select Register (INTC_PSR6)	16	R/W	8000h	21.5.6/608
6E	INTC Priority Select Register (INTC_PSR7)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
70	INTC Priority Select Register (INTC_PSR8)	16	R/W	8000h	21.5.6/608
72	INTC Priority Select Register (INTC_PSR9)	16	R/W	8000h	21.5.6/608
74	INTC Priority Select Register (INTC_PSR10)	16	R/W	8000h	21.5.6/608
76	INTC Priority Select Register (INTC_PSR11)	16	R/W	8000h	21.5.6/608
78	INTC Priority Select Register (INTC_PSR12)	16	R/W	8000h	21.5.6/608
7A	INTC Priority Select Register (INTC_PSR13)	16	R/W	8000h	21.5.6/608
7C	INTC Priority Select Register (INTC_PSR14)	16	R/W	8000h	21.5.6/608
7E	INTC Priority Select Register (INTC_PSR15)	16	R/W	8000h	21.5.6/608
80	INTC Priority Select Register (INTC_PSR16)	16	R/W	8000h	21.5.6/608
82	INTC Priority Select Register (INTC_PSR17)	16	R/W	8000h	21.5.6/608
84	INTC Priority Select Register (INTC_PSR18)	16	R/W	8000h	21.5.6/608
86	INTC Priority Select Register (INTC_PSR19)	16	R/W	8000h	21.5.6/608
88	INTC Priority Select Register (INTC_PSR20)	16	R/W	8000h	21.5.6/608
8A	INTC Priority Select Register (INTC_PSR21)	16	R/W	8000h	21.5.6/608
8C	INTC Priority Select Register (INTC_PSR22)	16	R/W	8000h	21.5.6/608
8E	INTC Priority Select Register (INTC_PSR23)	16	R/W	8000h	21.5.6/608
90	INTC Priority Select Register (INTC_PSR24)	16	R/W	8000h	21.5.6/608
92	INTC Priority Select Register (INTC_PSR25)	16	R/W	8000h	21.5.6/608
94	INTC Priority Select Register (INTC_PSR26)	16	R/W	8000h	21.5.6/608
96	INTC Priority Select Register (INTC_PSR27)	16	R/W	8000h	21.5.6/608
98	INTC Priority Select Register (INTC_PSR28)	16	R/W	8000h	21.5.6/608
9A	INTC Priority Select Register (INTC_PSR29)	16	R/W	8000h	21.5.6/608
9C	INTC Priority Select Register (INTC_PSR30)	16	R/W	8000h	21.5.6/608
9E	INTC Priority Select Register (INTC_PSR31)	16	R/W	8000h	21.5.6/608
A0	INTC Priority Select Register (INTC_PSR32)	16	R/W	8000h	21.5.6/608
A2	INTC Priority Select Register (INTC_PSR33)	16	R/W	8000h	21.5.6/608
A4	INTC Priority Select Register (INTC_PSR34)	16	R/W	8000h	21.5.6/608
A6	INTC Priority Select Register (INTC_PSR35)	16	R/W	8000h	21.5.6/608
A8	INTC Priority Select Register (INTC_PSR36)	16	R/W	8000h	21.5.6/608
AA	INTC Priority Select Register (INTC_PSR37)	16	R/W	8000h	21.5.6/608
AC	INTC Priority Select Register (INTC_PSR38)	16	R/W	8000h	21.5.6/608
AE	INTC Priority Select Register (INTC_PSR39)	16	R/W	8000h	21.5.6/608
B0	INTC Priority Select Register (INTC_PSR40)	16	R/W	8000h	21.5.6/608
B2	INTC Priority Select Register (INTC_PSR41)	16	R/W	8000h	21.5.6/608
B4	INTC Priority Select Register (INTC_PSR42)	16	R/W	8000h	21.5.6/608
B6	INTC Priority Select Register (INTC_PSR43)	16	R/W	8000h	21.5.6/608
B8	INTC Priority Select Register (INTC_PSR44)	16	R/W	8000h	21.5.6/608
BA	INTC Priority Select Register (INTC_PSR45)	16	R/W	8000h	21.5.6/608
BC	INTC Priority Select Register (INTC_PSR46)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BE	INTC Priority Select Register (INTC_PSR47)	16	R/W	8000h	21.5.6/608
C0	INTC Priority Select Register (INTC_PSR48)	16	R/W	8000h	21.5.6/608
C2	INTC Priority Select Register (INTC_PSR49)	16	R/W	8000h	21.5.6/608
C4	INTC Priority Select Register (INTC_PSR50)	16	R/W	8000h	21.5.6/608
C6	INTC Priority Select Register (INTC_PSR51)	16	R/W	8000h	21.5.6/608
C8	INTC Priority Select Register (INTC_PSR52)	16	R/W	8000h	21.5.6/608
CA	INTC Priority Select Register (INTC_PSR53)	16	R/W	8000h	21.5.6/608
CC	INTC Priority Select Register (INTC_PSR54)	16	R/W	8000h	21.5.6/608
CE	INTC Priority Select Register (INTC_PSR55)	16	R/W	8000h	21.5.6/608
D0	INTC Priority Select Register (INTC_PSR56)	16	R/W	8000h	21.5.6/608
D2	INTC Priority Select Register (INTC_PSR57)	16	R/W	8000h	21.5.6/608
D4	INTC Priority Select Register (INTC_PSR58)	16	R/W	8000h	21.5.6/608
D6	INTC Priority Select Register (INTC_PSR59)	16	R/W	8000h	21.5.6/608
D8	INTC Priority Select Register (INTC_PSR60)	16	R/W	8000h	21.5.6/608
DA	INTC Priority Select Register (INTC_PSR61)	16	R/W	8000h	21.5.6/608
DC	INTC Priority Select Register (INTC_PSR62)	16	R/W	8000h	21.5.6/608
DE	INTC Priority Select Register (INTC_PSR63)	16	R/W	8000h	21.5.6/608
E0	INTC Priority Select Register (INTC_PSR64)	16	R/W	8000h	21.5.6/608
E2	INTC Priority Select Register (INTC_PSR65)	16	R/W	8000h	21.5.6/608
E4	INTC Priority Select Register (INTC_PSR66)	16	R/W	8000h	21.5.6/608
E6	INTC Priority Select Register (INTC_PSR67)	16	R/W	8000h	21.5.6/608
E8	INTC Priority Select Register (INTC_PSR68)	16	R/W	8000h	21.5.6/608
EA	INTC Priority Select Register (INTC_PSR69)	16	R/W	8000h	21.5.6/608
EC	INTC Priority Select Register (INTC_PSR70)	16	R/W	8000h	21.5.6/608
EE	INTC Priority Select Register (INTC_PSR71)	16	R/W	8000h	21.5.6/608
F0	INTC Priority Select Register (INTC_PSR72)	16	R/W	8000h	21.5.6/608
F2	INTC Priority Select Register (INTC_PSR73)	16	R/W	8000h	21.5.6/608
F4	INTC Priority Select Register (INTC_PSR74)	16	R/W	8000h	21.5.6/608
F6	INTC Priority Select Register (INTC_PSR75)	16	R/W	8000h	21.5.6/608
F8	INTC Priority Select Register (INTC_PSR76)	16	R/W	8000h	21.5.6/608
FA	INTC Priority Select Register (INTC_PSR77)	16	R/W	8000h	21.5.6/608
FC	INTC Priority Select Register (INTC_PSR78)	16	R/W	8000h	21.5.6/608
FE	INTC Priority Select Register (INTC_PSR79)	16	R/W	8000h	21.5.6/608
100	INTC Priority Select Register (INTC_PSR80)	16	R/W	8000h	21.5.6/608
102	INTC Priority Select Register (INTC_PSR81)	16	R/W	8000h	21.5.6/608
104	INTC Priority Select Register (INTC_PSR82)	16	R/W	8000h	21.5.6/608
106	INTC Priority Select Register (INTC_PSR83)	16	R/W	8000h	21.5.6/608
108	INTC Priority Select Register (INTC_PSR84)	16	R/W	8000h	21.5.6/608
10A	INTC Priority Select Register (INTC_PSR85)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10C	INTC Priority Select Register (INTC_PSR86)	16	R/W	8000h	21.5.6/608
10E	INTC Priority Select Register (INTC_PSR87)	16	R/W	8000h	21.5.6/608
110	INTC Priority Select Register (INTC_PSR88)	16	R/W	8000h	21.5.6/608
112	INTC Priority Select Register (INTC_PSR89)	16	R/W	8000h	21.5.6/608
114	INTC Priority Select Register (INTC_PSR90)	16	R/W	8000h	21.5.6/608
116	INTC Priority Select Register (INTC_PSR91)	16	R/W	8000h	21.5.6/608
118	INTC Priority Select Register (INTC_PSR92)	16	R/W	8000h	21.5.6/608
11A	INTC Priority Select Register (INTC_PSR93)	16	R/W	8000h	21.5.6/608
11C	INTC Priority Select Register (INTC_PSR94)	16	R/W	8000h	21.5.6/608
11E	INTC Priority Select Register (INTC_PSR95)	16	R/W	8000h	21.5.6/608
120	INTC Priority Select Register (INTC_PSR96)	16	R/W	8000h	21.5.6/608
122	INTC Priority Select Register (INTC_PSR97)	16	R/W	8000h	21.5.6/608
124	INTC Priority Select Register (INTC_PSR98)	16	R/W	8000h	21.5.6/608
126	INTC Priority Select Register (INTC_PSR99)	16	R/W	8000h	21.5.6/608
128	INTC Priority Select Register (INTC_PSR100)	16	R/W	8000h	21.5.6/608
12A	INTC Priority Select Register (INTC_PSR101)	16	R/W	8000h	21.5.6/608
12C	INTC Priority Select Register (INTC_PSR102)	16	R/W	8000h	21.5.6/608
12E	INTC Priority Select Register (INTC_PSR103)	16	R/W	8000h	21.5.6/608
130	INTC Priority Select Register (INTC_PSR104)	16	R/W	8000h	21.5.6/608
132	INTC Priority Select Register (INTC_PSR105)	16	R/W	8000h	21.5.6/608
134	INTC Priority Select Register (INTC_PSR106)	16	R/W	8000h	21.5.6/608
136	INTC Priority Select Register (INTC_PSR107)	16	R/W	8000h	21.5.6/608
138	INTC Priority Select Register (INTC_PSR108)	16	R/W	8000h	21.5.6/608
13A	INTC Priority Select Register (INTC_PSR109)	16	R/W	8000h	21.5.6/608
13C	INTC Priority Select Register (INTC_PSR110)	16	R/W	8000h	21.5.6/608
13E	INTC Priority Select Register (INTC_PSR111)	16	R/W	8000h	21.5.6/608
140	INTC Priority Select Register (INTC_PSR112)	16	R/W	8000h	21.5.6/608
142	INTC Priority Select Register (INTC_PSR113)	16	R/W	8000h	21.5.6/608
144	INTC Priority Select Register (INTC_PSR114)	16	R/W	8000h	21.5.6/608
146	INTC Priority Select Register (INTC_PSR115)	16	R/W	8000h	21.5.6/608
148	INTC Priority Select Register (INTC_PSR116)	16	R/W	8000h	21.5.6/608
14A	INTC Priority Select Register (INTC_PSR117)	16	R/W	8000h	21.5.6/608
14C	INTC Priority Select Register (INTC_PSR118)	16	R/W	8000h	21.5.6/608
14E	INTC Priority Select Register (INTC_PSR119)	16	R/W	8000h	21.5.6/608
150	INTC Priority Select Register (INTC_PSR120)	16	R/W	8000h	21.5.6/608
152	INTC Priority Select Register (INTC_PSR121)	16	R/W	8000h	21.5.6/608
154	INTC Priority Select Register (INTC_PSR122)	16	R/W	8000h	21.5.6/608
156	INTC Priority Select Register (INTC_PSR123)	16	R/W	8000h	21.5.6/608
158	INTC Priority Select Register (INTC_PSR124)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15A	INTC Priority Select Register (INTC_PSR125)	16	R/W	8000h	21.5.6/608
15C	INTC Priority Select Register (INTC_PSR126)	16	R/W	8000h	21.5.6/608
15E	INTC Priority Select Register (INTC_PSR127)	16	R/W	8000h	21.5.6/608
160	INTC Priority Select Register (INTC_PSR128)	16	R/W	8000h	21.5.6/608
162	INTC Priority Select Register (INTC_PSR129)	16	R/W	8000h	21.5.6/608
164	INTC Priority Select Register (INTC_PSR130)	16	R/W	8000h	21.5.6/608
166	INTC Priority Select Register (INTC_PSR131)	16	R/W	8000h	21.5.6/608
168	INTC Priority Select Register (INTC_PSR132)	16	R/W	8000h	21.5.6/608
16A	INTC Priority Select Register (INTC_PSR133)	16	R/W	8000h	21.5.6/608
16C	INTC Priority Select Register (INTC_PSR134)	16	R/W	8000h	21.5.6/608
16E	INTC Priority Select Register (INTC_PSR135)	16	R/W	8000h	21.5.6/608
170	INTC Priority Select Register (INTC_PSR136)	16	R/W	8000h	21.5.6/608
172	INTC Priority Select Register (INTC_PSR137)	16	R/W	8000h	21.5.6/608
174	INTC Priority Select Register (INTC_PSR138)	16	R/W	8000h	21.5.6/608
176	INTC Priority Select Register (INTC_PSR139)	16	R/W	8000h	21.5.6/608
178	INTC Priority Select Register (INTC_PSR140)	16	R/W	8000h	21.5.6/608
17A	INTC Priority Select Register (INTC_PSR141)	16	R/W	8000h	21.5.6/608
17C	INTC Priority Select Register (INTC_PSR142)	16	R/W	8000h	21.5.6/608
17E	INTC Priority Select Register (INTC_PSR143)	16	R/W	8000h	21.5.6/608
180	INTC Priority Select Register (INTC_PSR144)	16	R/W	8000h	21.5.6/608
182	INTC Priority Select Register (INTC_PSR145)	16	R/W	8000h	21.5.6/608
184	INTC Priority Select Register (INTC_PSR146)	16	R/W	8000h	21.5.6/608
186	INTC Priority Select Register (INTC_PSR147)	16	R/W	8000h	21.5.6/608
188	INTC Priority Select Register (INTC_PSR148)	16	R/W	8000h	21.5.6/608
18A	INTC Priority Select Register (INTC_PSR149)	16	R/W	8000h	21.5.6/608
18C	INTC Priority Select Register (INTC_PSR150)	16	R/W	8000h	21.5.6/608
18E	INTC Priority Select Register (INTC_PSR151)	16	R/W	8000h	21.5.6/608
190	INTC Priority Select Register (INTC_PSR152)	16	R/W	8000h	21.5.6/608
192	INTC Priority Select Register (INTC_PSR153)	16	R/W	8000h	21.5.6/608
194	INTC Priority Select Register (INTC_PSR154)	16	R/W	8000h	21.5.6/608
196	INTC Priority Select Register (INTC_PSR155)	16	R/W	8000h	21.5.6/608
198	INTC Priority Select Register (INTC_PSR156)	16	R/W	8000h	21.5.6/608
19A	INTC Priority Select Register (INTC_PSR157)	16	R/W	8000h	21.5.6/608
19C	INTC Priority Select Register (INTC_PSR158)	16	R/W	8000h	21.5.6/608
19E	INTC Priority Select Register (INTC_PSR159)	16	R/W	8000h	21.5.6/608
1A0	INTC Priority Select Register (INTC_PSR160)	16	R/W	8000h	21.5.6/608
1A2	INTC Priority Select Register (INTC_PSR161)	16	R/W	8000h	21.5.6/608
1A4	INTC Priority Select Register (INTC_PSR162)	16	R/W	8000h	21.5.6/608
1A6	INTC Priority Select Register (INTC_PSR163)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A8	INTC Priority Select Register (INTC_PSR164)	16	R/W	8000h	21.5.6/608
1AA	INTC Priority Select Register (INTC_PSR165)	16	R/W	8000h	21.5.6/608
1AC	INTC Priority Select Register (INTC_PSR166)	16	R/W	8000h	21.5.6/608
1AE	INTC Priority Select Register (INTC_PSR167)	16	R/W	8000h	21.5.6/608
1B0	INTC Priority Select Register (INTC_PSR168)	16	R/W	8000h	21.5.6/608
1B2	INTC Priority Select Register (INTC_PSR169)	16	R/W	8000h	21.5.6/608
1B4	INTC Priority Select Register (INTC_PSR170)	16	R/W	8000h	21.5.6/608
1B6	INTC Priority Select Register (INTC_PSR171)	16	R/W	8000h	21.5.6/608
1B8	INTC Priority Select Register (INTC_PSR172)	16	R/W	8000h	21.5.6/608
1BA	INTC Priority Select Register (INTC_PSR173)	16	R/W	8000h	21.5.6/608
1BC	INTC Priority Select Register (INTC_PSR174)	16	R/W	8000h	21.5.6/608
1BE	INTC Priority Select Register (INTC_PSR175)	16	R/W	8000h	21.5.6/608
1C0	INTC Priority Select Register (INTC_PSR176)	16	R/W	8000h	21.5.6/608
1C2	INTC Priority Select Register (INTC_PSR177)	16	R/W	8000h	21.5.6/608
1C4	INTC Priority Select Register (INTC_PSR178)	16	R/W	8000h	21.5.6/608
1C6	INTC Priority Select Register (INTC_PSR179)	16	R/W	8000h	21.5.6/608
1C8	INTC Priority Select Register (INTC_PSR180)	16	R/W	8000h	21.5.6/608
1CA	INTC Priority Select Register (INTC_PSR181)	16	R/W	8000h	21.5.6/608
1CC	INTC Priority Select Register (INTC_PSR182)	16	R/W	8000h	21.5.6/608
1CE	INTC Priority Select Register (INTC_PSR183)	16	R/W	8000h	21.5.6/608
1D0	INTC Priority Select Register (INTC_PSR184)	16	R/W	8000h	21.5.6/608
1D2	INTC Priority Select Register (INTC_PSR185)	16	R/W	8000h	21.5.6/608
1D4	INTC Priority Select Register (INTC_PSR186)	16	R/W	8000h	21.5.6/608
1D6	INTC Priority Select Register (INTC_PSR187)	16	R/W	8000h	21.5.6/608
1D8	INTC Priority Select Register (INTC_PSR188)	16	R/W	8000h	21.5.6/608
1DA	INTC Priority Select Register (INTC_PSR189)	16	R/W	8000h	21.5.6/608
1DC	INTC Priority Select Register (INTC_PSR190)	16	R/W	8000h	21.5.6/608
1DE	INTC Priority Select Register (INTC_PSR191)	16	R/W	8000h	21.5.6/608
1E0	INTC Priority Select Register (INTC_PSR192)	16	R/W	8000h	21.5.6/608
1E2	INTC Priority Select Register (INTC_PSR193)	16	R/W	8000h	21.5.6/608
1E4	INTC Priority Select Register (INTC_PSR194)	16	R/W	8000h	21.5.6/608
1E6	INTC Priority Select Register (INTC_PSR195)	16	R/W	8000h	21.5.6/608
1E8	INTC Priority Select Register (INTC_PSR196)	16	R/W	8000h	21.5.6/608
1EA	INTC Priority Select Register (INTC_PSR197)	16	R/W	8000h	21.5.6/608
1EC	INTC Priority Select Register (INTC_PSR198)	16	R/W	8000h	21.5.6/608
1EE	INTC Priority Select Register (INTC_PSR199)	16	R/W	8000h	21.5.6/608
1F0	INTC Priority Select Register (INTC_PSR200)	16	R/W	8000h	21.5.6/608
1F2	INTC Priority Select Register (INTC_PSR201)	16	R/W	8000h	21.5.6/608
1F4	INTC Priority Select Register (INTC_PSR202)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1F6	INTC Priority Select Register (INTC_PSR203)	16	R/W	8000h	21.5.6/608
1F8	INTC Priority Select Register (INTC_PSR204)	16	R/W	8000h	21.5.6/608
1FA	INTC Priority Select Register (INTC_PSR205)	16	R/W	8000h	21.5.6/608
1FC	INTC Priority Select Register (INTC_PSR206)	16	R/W	8000h	21.5.6/608
1FE	INTC Priority Select Register (INTC_PSR207)	16	R/W	8000h	21.5.6/608
200	INTC Priority Select Register (INTC_PSR208)	16	R/W	8000h	21.5.6/608
202	INTC Priority Select Register (INTC_PSR209)	16	R/W	8000h	21.5.6/608
204	INTC Priority Select Register (INTC_PSR210)	16	R/W	8000h	21.5.6/608
206	INTC Priority Select Register (INTC_PSR211)	16	R/W	8000h	21.5.6/608
208	INTC Priority Select Register (INTC_PSR212)	16	R/W	8000h	21.5.6/608
20A	INTC Priority Select Register (INTC_PSR213)	16	R/W	8000h	21.5.6/608
20C	INTC Priority Select Register (INTC_PSR214)	16	R/W	8000h	21.5.6/608
20E	INTC Priority Select Register (INTC_PSR215)	16	R/W	8000h	21.5.6/608
210	INTC Priority Select Register (INTC_PSR216)	16	R/W	8000h	21.5.6/608
212	INTC Priority Select Register (INTC_PSR217)	16	R/W	8000h	21.5.6/608
214	INTC Priority Select Register (INTC_PSR218)	16	R/W	8000h	21.5.6/608
216	INTC Priority Select Register (INTC_PSR219)	16	R/W	8000h	21.5.6/608
218	INTC Priority Select Register (INTC_PSR220)	16	R/W	8000h	21.5.6/608
21A	INTC Priority Select Register (INTC_PSR221)	16	R/W	8000h	21.5.6/608
21C	INTC Priority Select Register (INTC_PSR222)	16	R/W	8000h	21.5.6/608
21E	INTC Priority Select Register (INTC_PSR223)	16	R/W	8000h	21.5.6/608
220	INTC Priority Select Register (INTC_PSR224)	16	R/W	8000h	21.5.6/608
222	INTC Priority Select Register (INTC_PSR225)	16	R/W	8000h	21.5.6/608
224	INTC Priority Select Register (INTC_PSR226)	16	R/W	8000h	21.5.6/608
226	INTC Priority Select Register (INTC_PSR227)	16	R/W	8000h	21.5.6/608
228	INTC Priority Select Register (INTC_PSR228)	16	R/W	8000h	21.5.6/608
22A	INTC Priority Select Register (INTC_PSR229)	16	R/W	8000h	21.5.6/608
22C	INTC Priority Select Register (INTC_PSR230)	16	R/W	8000h	21.5.6/608
22E	INTC Priority Select Register (INTC_PSR231)	16	R/W	8000h	21.5.6/608
230	INTC Priority Select Register (INTC_PSR232)	16	R/W	8000h	21.5.6/608
232	INTC Priority Select Register (INTC_PSR233)	16	R/W	8000h	21.5.6/608
234	INTC Priority Select Register (INTC_PSR234)	16	R/W	8000h	21.5.6/608
236	INTC Priority Select Register (INTC_PSR235)	16	R/W	8000h	21.5.6/608
238	INTC Priority Select Register (INTC_PSR236)	16	R/W	8000h	21.5.6/608
23A	INTC Priority Select Register (INTC_PSR237)	16	R/W	8000h	21.5.6/608
23C	INTC Priority Select Register (INTC_PSR238)	16	R/W	8000h	21.5.6/608
23E	INTC Priority Select Register (INTC_PSR239)	16	R/W	8000h	21.5.6/608
240	INTC Priority Select Register (INTC_PSR240)	16	R/W	8000h	21.5.6/608
242	INTC Priority Select Register (INTC_PSR241)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
244	INTC Priority Select Register (INTC_PSR242)	16	R/W	8000h	21.5.6/608
246	INTC Priority Select Register (INTC_PSR243)	16	R/W	8000h	21.5.6/608
248	INTC Priority Select Register (INTC_PSR244)	16	R/W	8000h	21.5.6/608
24A	INTC Priority Select Register (INTC_PSR245)	16	R/W	8000h	21.5.6/608
24C	INTC Priority Select Register (INTC_PSR246)	16	R/W	8000h	21.5.6/608
24E	INTC Priority Select Register (INTC_PSR247)	16	R/W	8000h	21.5.6/608
250	INTC Priority Select Register (INTC_PSR248)	16	R/W	8000h	21.5.6/608
252	INTC Priority Select Register (INTC_PSR249)	16	R/W	8000h	21.5.6/608
254	INTC Priority Select Register (INTC_PSR250)	16	R/W	8000h	21.5.6/608
256	INTC Priority Select Register (INTC_PSR251)	16	R/W	8000h	21.5.6/608
258	INTC Priority Select Register (INTC_PSR252)	16	R/W	8000h	21.5.6/608
25A	INTC Priority Select Register (INTC_PSR253)	16	R/W	8000h	21.5.6/608
25C	INTC Priority Select Register (INTC_PSR254)	16	R/W	8000h	21.5.6/608
25E	INTC Priority Select Register (INTC_PSR255)	16	R/W	8000h	21.5.6/608
260	INTC Priority Select Register (INTC_PSR256)	16	R/W	8000h	21.5.6/608
262	INTC Priority Select Register (INTC_PSR257)	16	R/W	8000h	21.5.6/608
264	INTC Priority Select Register (INTC_PSR258)	16	R/W	8000h	21.5.6/608
266	INTC Priority Select Register (INTC_PSR259)	16	R/W	8000h	21.5.6/608
268	INTC Priority Select Register (INTC_PSR260)	16	R/W	8000h	21.5.6/608
26A	INTC Priority Select Register (INTC_PSR261)	16	R/W	8000h	21.5.6/608
26C	INTC Priority Select Register (INTC_PSR262)	16	R/W	8000h	21.5.6/608
26E	INTC Priority Select Register (INTC_PSR263)	16	R/W	8000h	21.5.6/608
270	INTC Priority Select Register (INTC_PSR264)	16	R/W	8000h	21.5.6/608
272	INTC Priority Select Register (INTC_PSR265)	16	R/W	8000h	21.5.6/608
274	INTC Priority Select Register (INTC_PSR266)	16	R/W	8000h	21.5.6/608
276	INTC Priority Select Register (INTC_PSR267)	16	R/W	8000h	21.5.6/608
278	INTC Priority Select Register (INTC_PSR268)	16	R/W	8000h	21.5.6/608
27A	INTC Priority Select Register (INTC_PSR269)	16	R/W	8000h	21.5.6/608
27C	INTC Priority Select Register (INTC_PSR270)	16	R/W	8000h	21.5.6/608
27E	INTC Priority Select Register (INTC_PSR271)	16	R/W	8000h	21.5.6/608
280	INTC Priority Select Register (INTC_PSR272)	16	R/W	8000h	21.5.6/608
282	INTC Priority Select Register (INTC_PSR273)	16	R/W	8000h	21.5.6/608
284	INTC Priority Select Register (INTC_PSR274)	16	R/W	8000h	21.5.6/608
286	INTC Priority Select Register (INTC_PSR275)	16	R/W	8000h	21.5.6/608
288	INTC Priority Select Register (INTC_PSR276)	16	R/W	8000h	21.5.6/608
28A	INTC Priority Select Register (INTC_PSR277)	16	R/W	8000h	21.5.6/608
28C	INTC Priority Select Register (INTC_PSR278)	16	R/W	8000h	21.5.6/608
28E	INTC Priority Select Register (INTC_PSR279)	16	R/W	8000h	21.5.6/608
290	INTC Priority Select Register (INTC_PSR280)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
292	INTC Priority Select Register (INTC_PSR281)	16	R/W	8000h	21.5.6/608
294	INTC Priority Select Register (INTC_PSR282)	16	R/W	8000h	21.5.6/608
296	INTC Priority Select Register (INTC_PSR283)	16	R/W	8000h	21.5.6/608
298	INTC Priority Select Register (INTC_PSR284)	16	R/W	8000h	21.5.6/608
29A	INTC Priority Select Register (INTC_PSR285)	16	R/W	8000h	21.5.6/608
29C	INTC Priority Select Register (INTC_PSR286)	16	R/W	8000h	21.5.6/608
29E	INTC Priority Select Register (INTC_PSR287)	16	R/W	8000h	21.5.6/608
2A0	INTC Priority Select Register (INTC_PSR288)	16	R/W	8000h	21.5.6/608
2A2	INTC Priority Select Register (INTC_PSR289)	16	R/W	8000h	21.5.6/608
2A4	INTC Priority Select Register (INTC_PSR290)	16	R/W	8000h	21.5.6/608
2A6	INTC Priority Select Register (INTC_PSR291)	16	R/W	8000h	21.5.6/608
2A8	INTC Priority Select Register (INTC_PSR292)	16	R/W	8000h	21.5.6/608
2AA	INTC Priority Select Register (INTC_PSR293)	16	R/W	8000h	21.5.6/608
2AC	INTC Priority Select Register (INTC_PSR294)	16	R/W	8000h	21.5.6/608
2AE	INTC Priority Select Register (INTC_PSR295)	16	R/W	8000h	21.5.6/608
2B0	INTC Priority Select Register (INTC_PSR296)	16	R/W	8000h	21.5.6/608
2B2	INTC Priority Select Register (INTC_PSR297)	16	R/W	8000h	21.5.6/608
2B4	INTC Priority Select Register (INTC_PSR298)	16	R/W	8000h	21.5.6/608
2B6	INTC Priority Select Register (INTC_PSR299)	16	R/W	8000h	21.5.6/608
2B8	INTC Priority Select Register (INTC_PSR300)	16	R/W	8000h	21.5.6/608
2BA	INTC Priority Select Register (INTC_PSR301)	16	R/W	8000h	21.5.6/608
2BC	INTC Priority Select Register (INTC_PSR302)	16	R/W	8000h	21.5.6/608
2BE	INTC Priority Select Register (INTC_PSR303)	16	R/W	8000h	21.5.6/608
2C0	INTC Priority Select Register (INTC_PSR304)	16	R/W	8000h	21.5.6/608
2C2	INTC Priority Select Register (INTC_PSR305)	16	R/W	8000h	21.5.6/608
2C4	INTC Priority Select Register (INTC_PSR306)	16	R/W	8000h	21.5.6/608
2C6	INTC Priority Select Register (INTC_PSR307)	16	R/W	8000h	21.5.6/608
2C8	INTC Priority Select Register (INTC_PSR308)	16	R/W	8000h	21.5.6/608
2CA	INTC Priority Select Register (INTC_PSR309)	16	R/W	8000h	21.5.6/608
2CC	INTC Priority Select Register (INTC_PSR310)	16	R/W	8000h	21.5.6/608
2CE	INTC Priority Select Register (INTC_PSR311)	16	R/W	8000h	21.5.6/608
2D0	INTC Priority Select Register (INTC_PSR312)	16	R/W	8000h	21.5.6/608
2D2	INTC Priority Select Register (INTC_PSR313)	16	R/W	8000h	21.5.6/608
2D4	INTC Priority Select Register (INTC_PSR314)	16	R/W	8000h	21.5.6/608
2D6	INTC Priority Select Register (INTC_PSR315)	16	R/W	8000h	21.5.6/608
2D8	INTC Priority Select Register (INTC_PSR316)	16	R/W	8000h	21.5.6/608
2DA	INTC Priority Select Register (INTC_PSR317)	16	R/W	8000h	21.5.6/608
2DC	INTC Priority Select Register (INTC_PSR318)	16	R/W	8000h	21.5.6/608
2DE	INTC Priority Select Register (INTC_PSR319)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2E0	INTC Priority Select Register (INTC_PSR320)	16	R/W	8000h	21.5.6/608
2E2	INTC Priority Select Register (INTC_PSR321)	16	R/W	8000h	21.5.6/608
2E4	INTC Priority Select Register (INTC_PSR322)	16	R/W	8000h	21.5.6/608
2E6	INTC Priority Select Register (INTC_PSR323)	16	R/W	8000h	21.5.6/608
2E8	INTC Priority Select Register (INTC_PSR324)	16	R/W	8000h	21.5.6/608
2EA	INTC Priority Select Register (INTC_PSR325)	16	R/W	8000h	21.5.6/608
2EC	INTC Priority Select Register (INTC_PSR326)	16	R/W	8000h	21.5.6/608
2EE	INTC Priority Select Register (INTC_PSR327)	16	R/W	8000h	21.5.6/608
2F0	INTC Priority Select Register (INTC_PSR328)	16	R/W	8000h	21.5.6/608
2F2	INTC Priority Select Register (INTC_PSR329)	16	R/W	8000h	21.5.6/608
2F4	INTC Priority Select Register (INTC_PSR330)	16	R/W	8000h	21.5.6/608
2F6	INTC Priority Select Register (INTC_PSR331)	16	R/W	8000h	21.5.6/608
2F8	INTC Priority Select Register (INTC_PSR332)	16	R/W	8000h	21.5.6/608
2FA	INTC Priority Select Register (INTC_PSR333)	16	R/W	8000h	21.5.6/608
2FC	INTC Priority Select Register (INTC_PSR334)	16	R/W	8000h	21.5.6/608
2FE	INTC Priority Select Register (INTC_PSR335)	16	R/W	8000h	21.5.6/608
300	INTC Priority Select Register (INTC_PSR336)	16	R/W	8000h	21.5.6/608
302	INTC Priority Select Register (INTC_PSR337)	16	R/W	8000h	21.5.6/608
304	INTC Priority Select Register (INTC_PSR338)	16	R/W	8000h	21.5.6/608
306	INTC Priority Select Register (INTC_PSR339)	16	R/W	8000h	21.5.6/608
308	INTC Priority Select Register (INTC_PSR340)	16	R/W	8000h	21.5.6/608
30A	INTC Priority Select Register (INTC_PSR341)	16	R/W	8000h	21.5.6/608
30C	INTC Priority Select Register (INTC_PSR342)	16	R/W	8000h	21.5.6/608
30E	INTC Priority Select Register (INTC_PSR343)	16	R/W	8000h	21.5.6/608
310	INTC Priority Select Register (INTC_PSR344)	16	R/W	8000h	21.5.6/608
312	INTC Priority Select Register (INTC_PSR345)	16	R/W	8000h	21.5.6/608
314	INTC Priority Select Register (INTC_PSR346)	16	R/W	8000h	21.5.6/608
316	INTC Priority Select Register (INTC_PSR347)	16	R/W	8000h	21.5.6/608
318	INTC Priority Select Register (INTC_PSR348)	16	R/W	8000h	21.5.6/608
31A	INTC Priority Select Register (INTC_PSR349)	16	R/W	8000h	21.5.6/608
31C	INTC Priority Select Register (INTC_PSR350)	16	R/W	8000h	21.5.6/608
31E	INTC Priority Select Register (INTC_PSR351)	16	R/W	8000h	21.5.6/608
320	INTC Priority Select Register (INTC_PSR352)	16	R/W	8000h	21.5.6/608
322	INTC Priority Select Register (INTC_PSR353)	16	R/W	8000h	21.5.6/608
324	INTC Priority Select Register (INTC_PSR354)	16	R/W	8000h	21.5.6/608
326	INTC Priority Select Register (INTC_PSR355)	16	R/W	8000h	21.5.6/608
328	INTC Priority Select Register (INTC_PSR356)	16	R/W	8000h	21.5.6/608
32A	INTC Priority Select Register (INTC_PSR357)	16	R/W	8000h	21.5.6/608
32C	INTC Priority Select Register (INTC_PSR358)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
32E	INTC Priority Select Register (INTC_PSR359)	16	R/W	8000h	21.5.6/608
330	INTC Priority Select Register (INTC_PSR360)	16	R/W	8000h	21.5.6/608
332	INTC Priority Select Register (INTC_PSR361)	16	R/W	8000h	21.5.6/608
334	INTC Priority Select Register (INTC_PSR362)	16	R/W	8000h	21.5.6/608
336	INTC Priority Select Register (INTC_PSR363)	16	R/W	8000h	21.5.6/608
338	INTC Priority Select Register (INTC_PSR364)	16	R/W	8000h	21.5.6/608
33A	INTC Priority Select Register (INTC_PSR365)	16	R/W	8000h	21.5.6/608
33C	INTC Priority Select Register (INTC_PSR366)	16	R/W	8000h	21.5.6/608
33E	INTC Priority Select Register (INTC_PSR367)	16	R/W	8000h	21.5.6/608
340	INTC Priority Select Register (INTC_PSR368)	16	R/W	8000h	21.5.6/608
342	INTC Priority Select Register (INTC_PSR369)	16	R/W	8000h	21.5.6/608
344	INTC Priority Select Register (INTC_PSR370)	16	R/W	8000h	21.5.6/608
346	INTC Priority Select Register (INTC_PSR371)	16	R/W	8000h	21.5.6/608
348	INTC Priority Select Register (INTC_PSR372)	16	R/W	8000h	21.5.6/608
34A	INTC Priority Select Register (INTC_PSR373)	16	R/W	8000h	21.5.6/608
34C	INTC Priority Select Register (INTC_PSR374)	16	R/W	8000h	21.5.6/608
34E	INTC Priority Select Register (INTC_PSR375)	16	R/W	8000h	21.5.6/608
350	INTC Priority Select Register (INTC_PSR376)	16	R/W	8000h	21.5.6/608
352	INTC Priority Select Register (INTC_PSR377)	16	R/W	8000h	21.5.6/608
354	INTC Priority Select Register (INTC_PSR378)	16	R/W	8000h	21.5.6/608
356	INTC Priority Select Register (INTC_PSR379)	16	R/W	8000h	21.5.6/608
358	INTC Priority Select Register (INTC_PSR380)	16	R/W	8000h	21.5.6/608
35A	INTC Priority Select Register (INTC_PSR381)	16	R/W	8000h	21.5.6/608
35C	INTC Priority Select Register (INTC_PSR382)	16	R/W	8000h	21.5.6/608
35E	INTC Priority Select Register (INTC_PSR383)	16	R/W	8000h	21.5.6/608
360	INTC Priority Select Register (INTC_PSR384)	16	R/W	8000h	21.5.6/608
362	INTC Priority Select Register (INTC_PSR385)	16	R/W	8000h	21.5.6/608
364	INTC Priority Select Register (INTC_PSR386)	16	R/W	8000h	21.5.6/608
366	INTC Priority Select Register (INTC_PSR387)	16	R/W	8000h	21.5.6/608
368	INTC Priority Select Register (INTC_PSR388)	16	R/W	8000h	21.5.6/608
36A	INTC Priority Select Register (INTC_PSR389)	16	R/W	8000h	21.5.6/608
36C	INTC Priority Select Register (INTC_PSR390)	16	R/W	8000h	21.5.6/608
36E	INTC Priority Select Register (INTC_PSR391)	16	R/W	8000h	21.5.6/608
370	INTC Priority Select Register (INTC_PSR392)	16	R/W	8000h	21.5.6/608
372	INTC Priority Select Register (INTC_PSR393)	16	R/W	8000h	21.5.6/608
374	INTC Priority Select Register (INTC_PSR394)	16	R/W	8000h	21.5.6/608
376	INTC Priority Select Register (INTC_PSR395)	16	R/W	8000h	21.5.6/608
378	INTC Priority Select Register (INTC_PSR396)	16	R/W	8000h	21.5.6/608
37A	INTC Priority Select Register (INTC_PSR397)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
37C	INTC Priority Select Register (INTC_PSR398)	16	R/W	8000h	21.5.6/608
37E	INTC Priority Select Register (INTC_PSR399)	16	R/W	8000h	21.5.6/608
380	INTC Priority Select Register (INTC_PSR400)	16	R/W	8000h	21.5.6/608
382	INTC Priority Select Register (INTC_PSR401)	16	R/W	8000h	21.5.6/608
384	INTC Priority Select Register (INTC_PSR402)	16	R/W	8000h	21.5.6/608
386	INTC Priority Select Register (INTC_PSR403)	16	R/W	8000h	21.5.6/608
388	INTC Priority Select Register (INTC_PSR404)	16	R/W	8000h	21.5.6/608
38A	INTC Priority Select Register (INTC_PSR405)	16	R/W	8000h	21.5.6/608
38C	INTC Priority Select Register (INTC_PSR406)	16	R/W	8000h	21.5.6/608
38E	INTC Priority Select Register (INTC_PSR407)	16	R/W	8000h	21.5.6/608
390	INTC Priority Select Register (INTC_PSR408)	16	R/W	8000h	21.5.6/608
392	INTC Priority Select Register (INTC_PSR409)	16	R/W	8000h	21.5.6/608
394	INTC Priority Select Register (INTC_PSR410)	16	R/W	8000h	21.5.6/608
396	INTC Priority Select Register (INTC_PSR411)	16	R/W	8000h	21.5.6/608
398	INTC Priority Select Register (INTC_PSR412)	16	R/W	8000h	21.5.6/608
39A	INTC Priority Select Register (INTC_PSR413)	16	R/W	8000h	21.5.6/608
39C	INTC Priority Select Register (INTC_PSR414)	16	R/W	8000h	21.5.6/608
39E	INTC Priority Select Register (INTC_PSR415)	16	R/W	8000h	21.5.6/608
3A0	INTC Priority Select Register (INTC_PSR416)	16	R/W	8000h	21.5.6/608
3A2	INTC Priority Select Register (INTC_PSR417)	16	R/W	8000h	21.5.6/608
3A4	INTC Priority Select Register (INTC_PSR418)	16	R/W	8000h	21.5.6/608
3A6	INTC Priority Select Register (INTC_PSR419)	16	R/W	8000h	21.5.6/608
3A8	INTC Priority Select Register (INTC_PSR420)	16	R/W	8000h	21.5.6/608
3AA	INTC Priority Select Register (INTC_PSR421)	16	R/W	8000h	21.5.6/608
3AC	INTC Priority Select Register (INTC_PSR422)	16	R/W	8000h	21.5.6/608
3AE	INTC Priority Select Register (INTC_PSR423)	16	R/W	8000h	21.5.6/608
3B0	INTC Priority Select Register (INTC_PSR424)	16	R/W	8000h	21.5.6/608
3B2	INTC Priority Select Register (INTC_PSR425)	16	R/W	8000h	21.5.6/608
3B4	INTC Priority Select Register (INTC_PSR426)	16	R/W	8000h	21.5.6/608
3B6	INTC Priority Select Register (INTC_PSR427)	16	R/W	8000h	21.5.6/608
3B8	INTC Priority Select Register (INTC_PSR428)	16	R/W	8000h	21.5.6/608
3BA	INTC Priority Select Register (INTC_PSR429)	16	R/W	8000h	21.5.6/608
3BC	INTC Priority Select Register (INTC_PSR430)	16	R/W	8000h	21.5.6/608
3BE	INTC Priority Select Register (INTC_PSR431)	16	R/W	8000h	21.5.6/608
3C0	INTC Priority Select Register (INTC_PSR432)	16	R/W	8000h	21.5.6/608
3C2	INTC Priority Select Register (INTC_PSR433)	16	R/W	8000h	21.5.6/608
3C4	INTC Priority Select Register (INTC_PSR434)	16	R/W	8000h	21.5.6/608
3C6	INTC Priority Select Register (INTC_PSR435)	16	R/W	8000h	21.5.6/608
3C8	INTC Priority Select Register (INTC_PSR436)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3CA	INTC Priority Select Register (INTC_PSR437)	16	R/W	8000h	21.5.6/608
3CC	INTC Priority Select Register (INTC_PSR438)	16	R/W	8000h	21.5.6/608
3CE	INTC Priority Select Register (INTC_PSR439)	16	R/W	8000h	21.5.6/608
3D0	INTC Priority Select Register (INTC_PSR440)	16	R/W	8000h	21.5.6/608
3D2	INTC Priority Select Register (INTC_PSR441)	16	R/W	8000h	21.5.6/608
3D4	INTC Priority Select Register (INTC_PSR442)	16	R/W	8000h	21.5.6/608
3D6	INTC Priority Select Register (INTC_PSR443)	16	R/W	8000h	21.5.6/608
3D8	INTC Priority Select Register (INTC_PSR444)	16	R/W	8000h	21.5.6/608
3DA	INTC Priority Select Register (INTC_PSR445)	16	R/W	8000h	21.5.6/608
3DC	INTC Priority Select Register (INTC_PSR446)	16	R/W	8000h	21.5.6/608
3DE	INTC Priority Select Register (INTC_PSR447)	16	R/W	8000h	21.5.6/608
3E0	INTC Priority Select Register (INTC_PSR448)	16	R/W	8000h	21.5.6/608
3E2	INTC Priority Select Register (INTC_PSR449)	16	R/W	8000h	21.5.6/608
3E4	INTC Priority Select Register (INTC_PSR450)	16	R/W	8000h	21.5.6/608
3E6	INTC Priority Select Register (INTC_PSR451)	16	R/W	8000h	21.5.6/608
3E8	INTC Priority Select Register (INTC_PSR452)	16	R/W	8000h	21.5.6/608
3EA	INTC Priority Select Register (INTC_PSR453)	16	R/W	8000h	21.5.6/608
3EC	INTC Priority Select Register (INTC_PSR454)	16	R/W	8000h	21.5.6/608
3EE	INTC Priority Select Register (INTC_PSR455)	16	R/W	8000h	21.5.6/608
3F0	INTC Priority Select Register (INTC_PSR456)	16	R/W	8000h	21.5.6/608
3F2	INTC Priority Select Register (INTC_PSR457)	16	R/W	8000h	21.5.6/608
3F4	INTC Priority Select Register (INTC_PSR458)	16	R/W	8000h	21.5.6/608
3F6	INTC Priority Select Register (INTC_PSR459)	16	R/W	8000h	21.5.6/608
3F8	INTC Priority Select Register (INTC_PSR460)	16	R/W	8000h	21.5.6/608
3FA	INTC Priority Select Register (INTC_PSR461)	16	R/W	8000h	21.5.6/608
3FC	INTC Priority Select Register (INTC_PSR462)	16	R/W	8000h	21.5.6/608
3FE	INTC Priority Select Register (INTC_PSR463)	16	R/W	8000h	21.5.6/608
400	INTC Priority Select Register (INTC_PSR464)	16	R/W	8000h	21.5.6/608
402	INTC Priority Select Register (INTC_PSR465)	16	R/W	8000h	21.5.6/608
404	INTC Priority Select Register (INTC_PSR466)	16	R/W	8000h	21.5.6/608
406	INTC Priority Select Register (INTC_PSR467)	16	R/W	8000h	21.5.6/608
408	INTC Priority Select Register (INTC_PSR468)	16	R/W	8000h	21.5.6/608
40A	INTC Priority Select Register (INTC_PSR469)	16	R/W	8000h	21.5.6/608
40C	INTC Priority Select Register (INTC_PSR470)	16	R/W	8000h	21.5.6/608
40E	INTC Priority Select Register (INTC_PSR471)	16	R/W	8000h	21.5.6/608
410	INTC Priority Select Register (INTC_PSR472)	16	R/W	8000h	21.5.6/608
412	INTC Priority Select Register (INTC_PSR473)	16	R/W	8000h	21.5.6/608
414	INTC Priority Select Register (INTC_PSR474)	16	R/W	8000h	21.5.6/608
416	INTC Priority Select Register (INTC_PSR475)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
418	INTC Priority Select Register (INTC_PSR476)	16	R/W	8000h	21.5.6/608
41A	INTC Priority Select Register (INTC_PSR477)	16	R/W	8000h	21.5.6/608
41C	INTC Priority Select Register (INTC_PSR478)	16	R/W	8000h	21.5.6/608
41E	INTC Priority Select Register (INTC_PSR479)	16	R/W	8000h	21.5.6/608
420	INTC Priority Select Register (INTC_PSR480)	16	R/W	8000h	21.5.6/608
422	INTC Priority Select Register (INTC_PSR481)	16	R/W	8000h	21.5.6/608
424	INTC Priority Select Register (INTC_PSR482)	16	R/W	8000h	21.5.6/608
426	INTC Priority Select Register (INTC_PSR483)	16	R/W	8000h	21.5.6/608
428	INTC Priority Select Register (INTC_PSR484)	16	R/W	8000h	21.5.6/608
42A	INTC Priority Select Register (INTC_PSR485)	16	R/W	8000h	21.5.6/608
42C	INTC Priority Select Register (INTC_PSR486)	16	R/W	8000h	21.5.6/608
42E	INTC Priority Select Register (INTC_PSR487)	16	R/W	8000h	21.5.6/608
430	INTC Priority Select Register (INTC_PSR488)	16	R/W	8000h	21.5.6/608
432	INTC Priority Select Register (INTC_PSR489)	16	R/W	8000h	21.5.6/608
434	INTC Priority Select Register (INTC_PSR490)	16	R/W	8000h	21.5.6/608
436	INTC Priority Select Register (INTC_PSR491)	16	R/W	8000h	21.5.6/608
438	INTC Priority Select Register (INTC_PSR492)	16	R/W	8000h	21.5.6/608
43A	INTC Priority Select Register (INTC_PSR493)	16	R/W	8000h	21.5.6/608
43C	INTC Priority Select Register (INTC_PSR494)	16	R/W	8000h	21.5.6/608
43E	INTC Priority Select Register (INTC_PSR495)	16	R/W	8000h	21.5.6/608
440	INTC Priority Select Register (INTC_PSR496)	16	R/W	8000h	21.5.6/608
442	INTC Priority Select Register (INTC_PSR497)	16	R/W	8000h	21.5.6/608
444	INTC Priority Select Register (INTC_PSR498)	16	R/W	8000h	21.5.6/608
446	INTC Priority Select Register (INTC_PSR499)	16	R/W	8000h	21.5.6/608
448	INTC Priority Select Register (INTC_PSR500)	16	R/W	8000h	21.5.6/608
44A	INTC Priority Select Register (INTC_PSR501)	16	R/W	8000h	21.5.6/608
44C	INTC Priority Select Register (INTC_PSR502)	16	R/W	8000h	21.5.6/608
44E	INTC Priority Select Register (INTC_PSR503)	16	R/W	8000h	21.5.6/608
450	INTC Priority Select Register (INTC_PSR504)	16	R/W	8000h	21.5.6/608
452	INTC Priority Select Register (INTC_PSR505)	16	R/W	8000h	21.5.6/608
454	INTC Priority Select Register (INTC_PSR506)	16	R/W	8000h	21.5.6/608
456	INTC Priority Select Register (INTC_PSR507)	16	R/W	8000h	21.5.6/608
458	INTC Priority Select Register (INTC_PSR508)	16	R/W	8000h	21.5.6/608
45A	INTC Priority Select Register (INTC_PSR509)	16	R/W	8000h	21.5.6/608
45C	INTC Priority Select Register (INTC_PSR510)	16	R/W	8000h	21.5.6/608
45E	INTC Priority Select Register (INTC_PSR511)	16	R/W	8000h	21.5.6/608
460	INTC Priority Select Register (INTC_PSR512)	16	R/W	8000h	21.5.6/608
462	INTC Priority Select Register (INTC_PSR513)	16	R/W	8000h	21.5.6/608
464	INTC Priority Select Register (INTC_PSR514)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
466	INTC Priority Select Register (INTC_PSR515)	16	R/W	8000h	21.5.6/608
468	INTC Priority Select Register (INTC_PSR516)	16	R/W	8000h	21.5.6/608
46A	INTC Priority Select Register (INTC_PSR517)	16	R/W	8000h	21.5.6/608
46C	INTC Priority Select Register (INTC_PSR518)	16	R/W	8000h	21.5.6/608
46E	INTC Priority Select Register (INTC_PSR519)	16	R/W	8000h	21.5.6/608
470	INTC Priority Select Register (INTC_PSR520)	16	R/W	8000h	21.5.6/608
472	INTC Priority Select Register (INTC_PSR521)	16	R/W	8000h	21.5.6/608
474	INTC Priority Select Register (INTC_PSR522)	16	R/W	8000h	21.5.6/608
476	INTC Priority Select Register (INTC_PSR523)	16	R/W	8000h	21.5.6/608
478	INTC Priority Select Register (INTC_PSR524)	16	R/W	8000h	21.5.6/608
47A	INTC Priority Select Register (INTC_PSR525)	16	R/W	8000h	21.5.6/608
47C	INTC Priority Select Register (INTC_PSR526)	16	R/W	8000h	21.5.6/608
47E	INTC Priority Select Register (INTC_PSR527)	16	R/W	8000h	21.5.6/608
480	INTC Priority Select Register (INTC_PSR528)	16	R/W	8000h	21.5.6/608
482	INTC Priority Select Register (INTC_PSR529)	16	R/W	8000h	21.5.6/608
484	INTC Priority Select Register (INTC_PSR530)	16	R/W	8000h	21.5.6/608
486	INTC Priority Select Register (INTC_PSR531)	16	R/W	8000h	21.5.6/608
488	INTC Priority Select Register (INTC_PSR532)	16	R/W	8000h	21.5.6/608
48A	INTC Priority Select Register (INTC_PSR533)	16	R/W	8000h	21.5.6/608
48C	INTC Priority Select Register (INTC_PSR534)	16	R/W	8000h	21.5.6/608
48E	INTC Priority Select Register (INTC_PSR535)	16	R/W	8000h	21.5.6/608
490	INTC Priority Select Register (INTC_PSR536)	16	R/W	8000h	21.5.6/608
492	INTC Priority Select Register (INTC_PSR537)	16	R/W	8000h	21.5.6/608
494	INTC Priority Select Register (INTC_PSR538)	16	R/W	8000h	21.5.6/608
496	INTC Priority Select Register (INTC_PSR539)	16	R/W	8000h	21.5.6/608
498	INTC Priority Select Register (INTC_PSR540)	16	R/W	8000h	21.5.6/608
49A	INTC Priority Select Register (INTC_PSR541)	16	R/W	8000h	21.5.6/608
49C	INTC Priority Select Register (INTC_PSR542)	16	R/W	8000h	21.5.6/608
49E	INTC Priority Select Register (INTC_PSR543)	16	R/W	8000h	21.5.6/608
4A0	INTC Priority Select Register (INTC_PSR544)	16	R/W	8000h	21.5.6/608
4A2	INTC Priority Select Register (INTC_PSR545)	16	R/W	8000h	21.5.6/608
4A4	INTC Priority Select Register (INTC_PSR546)	16	R/W	8000h	21.5.6/608
4A6	INTC Priority Select Register (INTC_PSR547)	16	R/W	8000h	21.5.6/608
4A8	INTC Priority Select Register (INTC_PSR548)	16	R/W	8000h	21.5.6/608
4AA	INTC Priority Select Register (INTC_PSR549)	16	R/W	8000h	21.5.6/608
4AC	INTC Priority Select Register (INTC_PSR550)	16	R/W	8000h	21.5.6/608
4AE	INTC Priority Select Register (INTC_PSR551)	16	R/W	8000h	21.5.6/608
4B0	INTC Priority Select Register (INTC_PSR552)	16	R/W	8000h	21.5.6/608
4B2	INTC Priority Select Register (INTC_PSR553)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4B4	INTC Priority Select Register (INTC_PSR554)	16	R/W	8000h	21.5.6/608
4B6	INTC Priority Select Register (INTC_PSR555)	16	R/W	8000h	21.5.6/608
4B8	INTC Priority Select Register (INTC_PSR556)	16	R/W	8000h	21.5.6/608
4BA	INTC Priority Select Register (INTC_PSR557)	16	R/W	8000h	21.5.6/608
4BC	INTC Priority Select Register (INTC_PSR558)	16	R/W	8000h	21.5.6/608
4BE	INTC Priority Select Register (INTC_PSR559)	16	R/W	8000h	21.5.6/608
4C0	INTC Priority Select Register (INTC_PSR560)	16	R/W	8000h	21.5.6/608
4C2	INTC Priority Select Register (INTC_PSR561)	16	R/W	8000h	21.5.6/608
4C4	INTC Priority Select Register (INTC_PSR562)	16	R/W	8000h	21.5.6/608
4C6	INTC Priority Select Register (INTC_PSR563)	16	R/W	8000h	21.5.6/608
4C8	INTC Priority Select Register (INTC_PSR564)	16	R/W	8000h	21.5.6/608
4CA	INTC Priority Select Register (INTC_PSR565)	16	R/W	8000h	21.5.6/608
4CC	INTC Priority Select Register (INTC_PSR566)	16	R/W	8000h	21.5.6/608
4CE	INTC Priority Select Register (INTC_PSR567)	16	R/W	8000h	21.5.6/608
4D0	INTC Priority Select Register (INTC_PSR568)	16	R/W	8000h	21.5.6/608
4D2	INTC Priority Select Register (INTC_PSR569)	16	R/W	8000h	21.5.6/608
4D4	INTC Priority Select Register (INTC_PSR570)	16	R/W	8000h	21.5.6/608
4D6	INTC Priority Select Register (INTC_PSR571)	16	R/W	8000h	21.5.6/608
4D8	INTC Priority Select Register (INTC_PSR572)	16	R/W	8000h	21.5.6/608
4DA	INTC Priority Select Register (INTC_PSR573)	16	R/W	8000h	21.5.6/608
4DC	INTC Priority Select Register (INTC_PSR574)	16	R/W	8000h	21.5.6/608
4DE	INTC Priority Select Register (INTC_PSR575)	16	R/W	8000h	21.5.6/608
4E0	INTC Priority Select Register (INTC_PSR576)	16	R/W	8000h	21.5.6/608
4E2	INTC Priority Select Register (INTC_PSR577)	16	R/W	8000h	21.5.6/608
4E4	INTC Priority Select Register (INTC_PSR578)	16	R/W	8000h	21.5.6/608
4E6	INTC Priority Select Register (INTC_PSR579)	16	R/W	8000h	21.5.6/608
4E8	INTC Priority Select Register (INTC_PSR580)	16	R/W	8000h	21.5.6/608
4EA	INTC Priority Select Register (INTC_PSR581)	16	R/W	8000h	21.5.6/608
4EC	INTC Priority Select Register (INTC_PSR582)	16	R/W	8000h	21.5.6/608
4EE	INTC Priority Select Register (INTC_PSR583)	16	R/W	8000h	21.5.6/608
4F0	INTC Priority Select Register (INTC_PSR584)	16	R/W	8000h	21.5.6/608
4F2	INTC Priority Select Register (INTC_PSR585)	16	R/W	8000h	21.5.6/608
4F4	INTC Priority Select Register (INTC_PSR586)	16	R/W	8000h	21.5.6/608
4F6	INTC Priority Select Register (INTC_PSR587)	16	R/W	8000h	21.5.6/608
4F8	INTC Priority Select Register (INTC_PSR588)	16	R/W	8000h	21.5.6/608
4FA	INTC Priority Select Register (INTC_PSR589)	16	R/W	8000h	21.5.6/608
4FC	INTC Priority Select Register (INTC_PSR590)	16	R/W	8000h	21.5.6/608
4FE	INTC Priority Select Register (INTC_PSR591)	16	R/W	8000h	21.5.6/608
500	INTC Priority Select Register (INTC_PSR592)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
502	INTC Priority Select Register (INTC_PSR593)	16	R/W	8000h	21.5.6/608
504	INTC Priority Select Register (INTC_PSR594)	16	R/W	8000h	21.5.6/608
506	INTC Priority Select Register (INTC_PSR595)	16	R/W	8000h	21.5.6/608
508	INTC Priority Select Register (INTC_PSR596)	16	R/W	8000h	21.5.6/608
50A	INTC Priority Select Register (INTC_PSR597)	16	R/W	8000h	21.5.6/608
50C	INTC Priority Select Register (INTC_PSR598)	16	R/W	8000h	21.5.6/608
50E	INTC Priority Select Register (INTC_PSR599)	16	R/W	8000h	21.5.6/608
510	INTC Priority Select Register (INTC_PSR600)	16	R/W	8000h	21.5.6/608
512	INTC Priority Select Register (INTC_PSR601)	16	R/W	8000h	21.5.6/608
514	INTC Priority Select Register (INTC_PSR602)	16	R/W	8000h	21.5.6/608
516	INTC Priority Select Register (INTC_PSR603)	16	R/W	8000h	21.5.6/608
518	INTC Priority Select Register (INTC_PSR604)	16	R/W	8000h	21.5.6/608
51A	INTC Priority Select Register (INTC_PSR605)	16	R/W	8000h	21.5.6/608
51C	INTC Priority Select Register (INTC_PSR606)	16	R/W	8000h	21.5.6/608
51E	INTC Priority Select Register (INTC_PSR607)	16	R/W	8000h	21.5.6/608
520	INTC Priority Select Register (INTC_PSR608)	16	R/W	8000h	21.5.6/608
522	INTC Priority Select Register (INTC_PSR609)	16	R/W	8000h	21.5.6/608
524	INTC Priority Select Register (INTC_PSR610)	16	R/W	8000h	21.5.6/608
526	INTC Priority Select Register (INTC_PSR611)	16	R/W	8000h	21.5.6/608
528	INTC Priority Select Register (INTC_PSR612)	16	R/W	8000h	21.5.6/608
52A	INTC Priority Select Register (INTC_PSR613)	16	R/W	8000h	21.5.6/608
52C	INTC Priority Select Register (INTC_PSR614)	16	R/W	8000h	21.5.6/608
52E	INTC Priority Select Register (INTC_PSR615)	16	R/W	8000h	21.5.6/608
530	INTC Priority Select Register (INTC_PSR616)	16	R/W	8000h	21.5.6/608
532	INTC Priority Select Register (INTC_PSR617)	16	R/W	8000h	21.5.6/608
534	INTC Priority Select Register (INTC_PSR618)	16	R/W	8000h	21.5.6/608
536	INTC Priority Select Register (INTC_PSR619)	16	R/W	8000h	21.5.6/608
538	INTC Priority Select Register (INTC_PSR620)	16	R/W	8000h	21.5.6/608
53A	INTC Priority Select Register (INTC_PSR621)	16	R/W	8000h	21.5.6/608
53C	INTC Priority Select Register (INTC_PSR622)	16	R/W	8000h	21.5.6/608
53E	INTC Priority Select Register (INTC_PSR623)	16	R/W	8000h	21.5.6/608
540	INTC Priority Select Register (INTC_PSR624)	16	R/W	8000h	21.5.6/608
542	INTC Priority Select Register (INTC_PSR625)	16	R/W	8000h	21.5.6/608
544	INTC Priority Select Register (INTC_PSR626)	16	R/W	8000h	21.5.6/608
546	INTC Priority Select Register (INTC_PSR627)	16	R/W	8000h	21.5.6/608
548	INTC Priority Select Register (INTC_PSR628)	16	R/W	8000h	21.5.6/608
54A	INTC Priority Select Register (INTC_PSR629)	16	R/W	8000h	21.5.6/608
54C	INTC Priority Select Register (INTC_PSR630)	16	R/W	8000h	21.5.6/608
54E	INTC Priority Select Register (INTC_PSR631)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
550	INTC Priority Select Register (INTC_PSR632)	16	R/W	8000h	21.5.6/608
552	INTC Priority Select Register (INTC_PSR633)	16	R/W	8000h	21.5.6/608
554	INTC Priority Select Register (INTC_PSR634)	16	R/W	8000h	21.5.6/608
556	INTC Priority Select Register (INTC_PSR635)	16	R/W	8000h	21.5.6/608
558	INTC Priority Select Register (INTC_PSR636)	16	R/W	8000h	21.5.6/608
55A	INTC Priority Select Register (INTC_PSR637)	16	R/W	8000h	21.5.6/608
55C	INTC Priority Select Register (INTC_PSR638)	16	R/W	8000h	21.5.6/608
55E	INTC Priority Select Register (INTC_PSR639)	16	R/W	8000h	21.5.6/608
560	INTC Priority Select Register (INTC_PSR640)	16	R/W	8000h	21.5.6/608
562	INTC Priority Select Register (INTC_PSR641)	16	R/W	8000h	21.5.6/608
564	INTC Priority Select Register (INTC_PSR642)	16	R/W	8000h	21.5.6/608
566	INTC Priority Select Register (INTC_PSR643)	16	R/W	8000h	21.5.6/608
568	INTC Priority Select Register (INTC_PSR644)	16	R/W	8000h	21.5.6/608
56A	INTC Priority Select Register (INTC_PSR645)	16	R/W	8000h	21.5.6/608
56C	INTC Priority Select Register (INTC_PSR646)	16	R/W	8000h	21.5.6/608
56E	INTC Priority Select Register (INTC_PSR647)	16	R/W	8000h	21.5.6/608
570	INTC Priority Select Register (INTC_PSR648)	16	R/W	8000h	21.5.6/608
572	INTC Priority Select Register (INTC_PSR649)	16	R/W	8000h	21.5.6/608
574	INTC Priority Select Register (INTC_PSR650)	16	R/W	8000h	21.5.6/608
576	INTC Priority Select Register (INTC_PSR651)	16	R/W	8000h	21.5.6/608
578	INTC Priority Select Register (INTC_PSR652)	16	R/W	8000h	21.5.6/608
57A	INTC Priority Select Register (INTC_PSR653)	16	R/W	8000h	21.5.6/608
57C	INTC Priority Select Register (INTC_PSR654)	16	R/W	8000h	21.5.6/608
57E	INTC Priority Select Register (INTC_PSR655)	16	R/W	8000h	21.5.6/608
580	INTC Priority Select Register (INTC_PSR656)	16	R/W	8000h	21.5.6/608
582	INTC Priority Select Register (INTC_PSR657)	16	R/W	8000h	21.5.6/608
584	INTC Priority Select Register (INTC_PSR658)	16	R/W	8000h	21.5.6/608
586	INTC Priority Select Register (INTC_PSR659)	16	R/W	8000h	21.5.6/608
588	INTC Priority Select Register (INTC_PSR660)	16	R/W	8000h	21.5.6/608
58A	INTC Priority Select Register (INTC_PSR661)	16	R/W	8000h	21.5.6/608
58C	INTC Priority Select Register (INTC_PSR662)	16	R/W	8000h	21.5.6/608
58E	INTC Priority Select Register (INTC_PSR663)	16	R/W	8000h	21.5.6/608
590	INTC Priority Select Register (INTC_PSR664)	16	R/W	8000h	21.5.6/608
592	INTC Priority Select Register (INTC_PSR665)	16	R/W	8000h	21.5.6/608
594	INTC Priority Select Register (INTC_PSR666)	16	R/W	8000h	21.5.6/608
596	INTC Priority Select Register (INTC_PSR667)	16	R/W	8000h	21.5.6/608
598	INTC Priority Select Register (INTC_PSR668)	16	R/W	8000h	21.5.6/608
59A	INTC Priority Select Register (INTC_PSR669)	16	R/W	8000h	21.5.6/608
59C	INTC Priority Select Register (INTC_PSR670)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
59E	INTC Priority Select Register (INTC_PSR671)	16	R/W	8000h	21.5.6/608
5A0	INTC Priority Select Register (INTC_PSR672)	16	R/W	8000h	21.5.6/608
5A2	INTC Priority Select Register (INTC_PSR673)	16	R/W	8000h	21.5.6/608
5A4	INTC Priority Select Register (INTC_PSR674)	16	R/W	8000h	21.5.6/608
5A6	INTC Priority Select Register (INTC_PSR675)	16	R/W	8000h	21.5.6/608
5A8	INTC Priority Select Register (INTC_PSR676)	16	R/W	8000h	21.5.6/608
5AA	INTC Priority Select Register (INTC_PSR677)	16	R/W	8000h	21.5.6/608
5AC	INTC Priority Select Register (INTC_PSR678)	16	R/W	8000h	21.5.6/608
5AE	INTC Priority Select Register (INTC_PSR679)	16	R/W	8000h	21.5.6/608
5B0	INTC Priority Select Register (INTC_PSR680)	16	R/W	8000h	21.5.6/608
5B2	INTC Priority Select Register (INTC_PSR681)	16	R/W	8000h	21.5.6/608
5B4	INTC Priority Select Register (INTC_PSR682)	16	R/W	8000h	21.5.6/608
5B6	INTC Priority Select Register (INTC_PSR683)	16	R/W	8000h	21.5.6/608
5B8	INTC Priority Select Register (INTC_PSR684)	16	R/W	8000h	21.5.6/608
5BA	INTC Priority Select Register (INTC_PSR685)	16	R/W	8000h	21.5.6/608
5BC	INTC Priority Select Register (INTC_PSR686)	16	R/W	8000h	21.5.6/608
5BE	INTC Priority Select Register (INTC_PSR687)	16	R/W	8000h	21.5.6/608
5C0	INTC Priority Select Register (INTC_PSR688)	16	R/W	8000h	21.5.6/608
5C2	INTC Priority Select Register (INTC_PSR689)	16	R/W	8000h	21.5.6/608
5C4	INTC Priority Select Register (INTC_PSR690)	16	R/W	8000h	21.5.6/608
5C6	INTC Priority Select Register (INTC_PSR691)	16	R/W	8000h	21.5.6/608
5C8	INTC Priority Select Register (INTC_PSR692)	16	R/W	8000h	21.5.6/608
5CA	INTC Priority Select Register (INTC_PSR693)	16	R/W	8000h	21.5.6/608
5CC	INTC Priority Select Register (INTC_PSR694)	16	R/W	8000h	21.5.6/608
5CE	INTC Priority Select Register (INTC_PSR695)	16	R/W	8000h	21.5.6/608
5D0	INTC Priority Select Register (INTC_PSR696)	16	R/W	8000h	21.5.6/608
5D2	INTC Priority Select Register (INTC_PSR697)	16	R/W	8000h	21.5.6/608
5D4	INTC Priority Select Register (INTC_PSR698)	16	R/W	8000h	21.5.6/608
5D6	INTC Priority Select Register (INTC_PSR699)	16	R/W	8000h	21.5.6/608
5D8	INTC Priority Select Register (INTC_PSR700)	16	R/W	8000h	21.5.6/608
5DA	INTC Priority Select Register (INTC_PSR701)	16	R/W	8000h	21.5.6/608
5DC	INTC Priority Select Register (INTC_PSR702)	16	R/W	8000h	21.5.6/608
5DE	INTC Priority Select Register (INTC_PSR703)	16	R/W	8000h	21.5.6/608
5E0	INTC Priority Select Register (INTC_PSR704)	16	R/W	8000h	21.5.6/608
5E2	INTC Priority Select Register (INTC_PSR705)	16	R/W	8000h	21.5.6/608
5E4	INTC Priority Select Register (INTC_PSR706)	16	R/W	8000h	21.5.6/608
5E6	INTC Priority Select Register (INTC_PSR707)	16	R/W	8000h	21.5.6/608
5E8	INTC Priority Select Register (INTC_PSR708)	16	R/W	8000h	21.5.6/608
5EA	INTC Priority Select Register (INTC_PSR709)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5EC	INTC Priority Select Register (INTC_PSR710)	16	R/W	8000h	21.5.6/608
5EE	INTC Priority Select Register (INTC_PSR711)	16	R/W	8000h	21.5.6/608
5F0	INTC Priority Select Register (INTC_PSR712)	16	R/W	8000h	21.5.6/608
5F2	INTC Priority Select Register (INTC_PSR713)	16	R/W	8000h	21.5.6/608
5F4	INTC Priority Select Register (INTC_PSR714)	16	R/W	8000h	21.5.6/608
5F6	INTC Priority Select Register (INTC_PSR715)	16	R/W	8000h	21.5.6/608
5F8	INTC Priority Select Register (INTC_PSR716)	16	R/W	8000h	21.5.6/608
5FA	INTC Priority Select Register (INTC_PSR717)	16	R/W	8000h	21.5.6/608
5FC	INTC Priority Select Register (INTC_PSR718)	16	R/W	8000h	21.5.6/608
5FE	INTC Priority Select Register (INTC_PSR719)	16	R/W	8000h	21.5.6/608
600	INTC Priority Select Register (INTC_PSR720)	16	R/W	8000h	21.5.6/608
602	INTC Priority Select Register (INTC_PSR721)	16	R/W	8000h	21.5.6/608
604	INTC Priority Select Register (INTC_PSR722)	16	R/W	8000h	21.5.6/608
606	INTC Priority Select Register (INTC_PSR723)	16	R/W	8000h	21.5.6/608
608	INTC Priority Select Register (INTC_PSR724)	16	R/W	8000h	21.5.6/608
60A	INTC Priority Select Register (INTC_PSR725)	16	R/W	8000h	21.5.6/608
60C	INTC Priority Select Register (INTC_PSR726)	16	R/W	8000h	21.5.6/608
60E	INTC Priority Select Register (INTC_PSR727)	16	R/W	8000h	21.5.6/608
610	INTC Priority Select Register (INTC_PSR728)	16	R/W	8000h	21.5.6/608
612	INTC Priority Select Register (INTC_PSR729)	16	R/W	8000h	21.5.6/608
614	INTC Priority Select Register (INTC_PSR730)	16	R/W	8000h	21.5.6/608
616	INTC Priority Select Register (INTC_PSR731)	16	R/W	8000h	21.5.6/608
618	INTC Priority Select Register (INTC_PSR732)	16	R/W	8000h	21.5.6/608
61A	INTC Priority Select Register (INTC_PSR733)	16	R/W	8000h	21.5.6/608
61C	INTC Priority Select Register (INTC_PSR734)	16	R/W	8000h	21.5.6/608
61E	INTC Priority Select Register (INTC_PSR735)	16	R/W	8000h	21.5.6/608
620	INTC Priority Select Register (INTC_PSR736)	16	R/W	8000h	21.5.6/608
622	INTC Priority Select Register (INTC_PSR737)	16	R/W	8000h	21.5.6/608
624	INTC Priority Select Register (INTC_PSR738)	16	R/W	8000h	21.5.6/608
626	INTC Priority Select Register (INTC_PSR739)	16	R/W	8000h	21.5.6/608
628	INTC Priority Select Register (INTC_PSR740)	16	R/W	8000h	21.5.6/608
62A	INTC Priority Select Register (INTC_PSR741)	16	R/W	8000h	21.5.6/608
62C	INTC Priority Select Register (INTC_PSR742)	16	R/W	8000h	21.5.6/608
62E	INTC Priority Select Register (INTC_PSR743)	16	R/W	8000h	21.5.6/608
630	INTC Priority Select Register (INTC_PSR744)	16	R/W	8000h	21.5.6/608
632	INTC Priority Select Register (INTC_PSR745)	16	R/W	8000h	21.5.6/608
634	INTC Priority Select Register (INTC_PSR746)	16	R/W	8000h	21.5.6/608
636	INTC Priority Select Register (INTC_PSR747)	16	R/W	8000h	21.5.6/608
638	INTC Priority Select Register (INTC_PSR748)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63A	INTC Priority Select Register (INTC_PSR749)	16	R/W	8000h	21.5.6/608
63C	INTC Priority Select Register (INTC_PSR750)	16	R/W	8000h	21.5.6/608
63E	INTC Priority Select Register (INTC_PSR751)	16	R/W	8000h	21.5.6/608
640	INTC Priority Select Register (INTC_PSR752)	16	R/W	8000h	21.5.6/608
642	INTC Priority Select Register (INTC_PSR753)	16	R/W	8000h	21.5.6/608
644	INTC Priority Select Register (INTC_PSR754)	16	R/W	8000h	21.5.6/608
646	INTC Priority Select Register (INTC_PSR755)	16	R/W	8000h	21.5.6/608
648	INTC Priority Select Register (INTC_PSR756)	16	R/W	8000h	21.5.6/608
64A	INTC Priority Select Register (INTC_PSR757)	16	R/W	8000h	21.5.6/608
64C	INTC Priority Select Register (INTC_PSR758)	16	R/W	8000h	21.5.6/608
64E	INTC Priority Select Register (INTC_PSR759)	16	R/W	8000h	21.5.6/608
650	INTC Priority Select Register (INTC_PSR760)	16	R/W	8000h	21.5.6/608
652	INTC Priority Select Register (INTC_PSR761)	16	R/W	8000h	21.5.6/608
654	INTC Priority Select Register (INTC_PSR762)	16	R/W	8000h	21.5.6/608
656	INTC Priority Select Register (INTC_PSR763)	16	R/W	8000h	21.5.6/608
658	INTC Priority Select Register (INTC_PSR764)	16	R/W	8000h	21.5.6/608
65A	INTC Priority Select Register (INTC_PSR765)	16	R/W	8000h	21.5.6/608
65C	INTC Priority Select Register (INTC_PSR766)	16	R/W	8000h	21.5.6/608
65E	INTC Priority Select Register (INTC_PSR767)	16	R/W	8000h	21.5.6/608
660	INTC Priority Select Register (INTC_PSR768)	16	R/W	8000h	21.5.6/608
662	INTC Priority Select Register (INTC_PSR769)	16	R/W	8000h	21.5.6/608
664	INTC Priority Select Register (INTC_PSR770)	16	R/W	8000h	21.5.6/608
666	INTC Priority Select Register (INTC_PSR771)	16	R/W	8000h	21.5.6/608
668	INTC Priority Select Register (INTC_PSR772)	16	R/W	8000h	21.5.6/608
66A	INTC Priority Select Register (INTC_PSR773)	16	R/W	8000h	21.5.6/608
66C	INTC Priority Select Register (INTC_PSR774)	16	R/W	8000h	21.5.6/608
66E	INTC Priority Select Register (INTC_PSR775)	16	R/W	8000h	21.5.6/608
670	INTC Priority Select Register (INTC_PSR776)	16	R/W	8000h	21.5.6/608
672	INTC Priority Select Register (INTC_PSR777)	16	R/W	8000h	21.5.6/608
674	INTC Priority Select Register (INTC_PSR778)	16	R/W	8000h	21.5.6/608
676	INTC Priority Select Register (INTC_PSR779)	16	R/W	8000h	21.5.6/608
678	INTC Priority Select Register (INTC_PSR780)	16	R/W	8000h	21.5.6/608
67A	INTC Priority Select Register (INTC_PSR781)	16	R/W	8000h	21.5.6/608
67C	INTC Priority Select Register (INTC_PSR782)	16	R/W	8000h	21.5.6/608
67E	INTC Priority Select Register (INTC_PSR783)	16	R/W	8000h	21.5.6/608
680	INTC Priority Select Register (INTC_PSR784)	16	R/W	8000h	21.5.6/608
682	INTC Priority Select Register (INTC_PSR785)	16	R/W	8000h	21.5.6/608
684	INTC Priority Select Register (INTC_PSR786)	16	R/W	8000h	21.5.6/608
686	INTC Priority Select Register (INTC_PSR787)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
688	INTC Priority Select Register (INTC_PSR788)	16	R/W	8000h	21.5.6/608
68A	INTC Priority Select Register (INTC_PSR789)	16	R/W	8000h	21.5.6/608
68C	INTC Priority Select Register (INTC_PSR790)	16	R/W	8000h	21.5.6/608
68E	INTC Priority Select Register (INTC_PSR791)	16	R/W	8000h	21.5.6/608
690	INTC Priority Select Register (INTC_PSR792)	16	R/W	8000h	21.5.6/608
692	INTC Priority Select Register (INTC_PSR793)	16	R/W	8000h	21.5.6/608
694	INTC Priority Select Register (INTC_PSR794)	16	R/W	8000h	21.5.6/608
696	INTC Priority Select Register (INTC_PSR795)	16	R/W	8000h	21.5.6/608
698	INTC Priority Select Register (INTC_PSR796)	16	R/W	8000h	21.5.6/608
69A	INTC Priority Select Register (INTC_PSR797)	16	R/W	8000h	21.5.6/608
69C	INTC Priority Select Register (INTC_PSR798)	16	R/W	8000h	21.5.6/608
69E	INTC Priority Select Register (INTC_PSR799)	16	R/W	8000h	21.5.6/608
6A0	INTC Priority Select Register (INTC_PSR800)	16	R/W	8000h	21.5.6/608
6A2	INTC Priority Select Register (INTC_PSR801)	16	R/W	8000h	21.5.6/608
6A4	INTC Priority Select Register (INTC_PSR802)	16	R/W	8000h	21.5.6/608
6A6	INTC Priority Select Register (INTC_PSR803)	16	R/W	8000h	21.5.6/608
6A8	INTC Priority Select Register (INTC_PSR804)	16	R/W	8000h	21.5.6/608
6AA	INTC Priority Select Register (INTC_PSR805)	16	R/W	8000h	21.5.6/608
6AC	INTC Priority Select Register (INTC_PSR806)	16	R/W	8000h	21.5.6/608
6AE	INTC Priority Select Register (INTC_PSR807)	16	R/W	8000h	21.5.6/608
6B0	INTC Priority Select Register (INTC_PSR808)	16	R/W	8000h	21.5.6/608
6B2	INTC Priority Select Register (INTC_PSR809)	16	R/W	8000h	21.5.6/608
6B4	INTC Priority Select Register (INTC_PSR810)	16	R/W	8000h	21.5.6/608
6B6	INTC Priority Select Register (INTC_PSR811)	16	R/W	8000h	21.5.6/608
6B8	INTC Priority Select Register (INTC_PSR812)	16	R/W	8000h	21.5.6/608
6BA	INTC Priority Select Register (INTC_PSR813)	16	R/W	8000h	21.5.6/608
6BC	INTC Priority Select Register (INTC_PSR814)	16	R/W	8000h	21.5.6/608
6BE	INTC Priority Select Register (INTC_PSR815)	16	R/W	8000h	21.5.6/608
6C0	INTC Priority Select Register (INTC_PSR816)	16	R/W	8000h	21.5.6/608
6C2	INTC Priority Select Register (INTC_PSR817)	16	R/W	8000h	21.5.6/608
6C4	INTC Priority Select Register (INTC_PSR818)	16	R/W	8000h	21.5.6/608
6C6	INTC Priority Select Register (INTC_PSR819)	16	R/W	8000h	21.5.6/608
6C8	INTC Priority Select Register (INTC_PSR820)	16	R/W	8000h	21.5.6/608
6CA	INTC Priority Select Register (INTC_PSR821)	16	R/W	8000h	21.5.6/608
6CC	INTC Priority Select Register (INTC_PSR822)	16	R/W	8000h	21.5.6/608
6CE	INTC Priority Select Register (INTC_PSR823)	16	R/W	8000h	21.5.6/608
6D0	INTC Priority Select Register (INTC_PSR824)	16	R/W	8000h	21.5.6/608
6D2	INTC Priority Select Register (INTC_PSR825)	16	R/W	8000h	21.5.6/608
6D4	INTC Priority Select Register (INTC_PSR826)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6D6	INTC Priority Select Register (INTC_PSR827)	16	R/W	8000h	21.5.6/608
6D8	INTC Priority Select Register (INTC_PSR828)	16	R/W	8000h	21.5.6/608
6DA	INTC Priority Select Register (INTC_PSR829)	16	R/W	8000h	21.5.6/608
6DC	INTC Priority Select Register (INTC_PSR830)	16	R/W	8000h	21.5.6/608
6DE	INTC Priority Select Register (INTC_PSR831)	16	R/W	8000h	21.5.6/608
6E0	INTC Priority Select Register (INTC_PSR832)	16	R/W	8000h	21.5.6/608
6E2	INTC Priority Select Register (INTC_PSR833)	16	R/W	8000h	21.5.6/608
6E4	INTC Priority Select Register (INTC_PSR834)	16	R/W	8000h	21.5.6/608
6E6	INTC Priority Select Register (INTC_PSR835)	16	R/W	8000h	21.5.6/608
6E8	INTC Priority Select Register (INTC_PSR836)	16	R/W	8000h	21.5.6/608
6EA	INTC Priority Select Register (INTC_PSR837)	16	R/W	8000h	21.5.6/608
6EC	INTC Priority Select Register (INTC_PSR838)	16	R/W	8000h	21.5.6/608
6EE	INTC Priority Select Register (INTC_PSR839)	16	R/W	8000h	21.5.6/608
6F0	INTC Priority Select Register (INTC_PSR840)	16	R/W	8000h	21.5.6/608
6F2	INTC Priority Select Register (INTC_PSR841)	16	R/W	8000h	21.5.6/608
6F4	INTC Priority Select Register (INTC_PSR842)	16	R/W	8000h	21.5.6/608
6F6	INTC Priority Select Register (INTC_PSR843)	16	R/W	8000h	21.5.6/608
6F8	INTC Priority Select Register (INTC_PSR844)	16	R/W	8000h	21.5.6/608
6FA	INTC Priority Select Register (INTC_PSR845)	16	R/W	8000h	21.5.6/608
6FC	INTC Priority Select Register (INTC_PSR846)	16	R/W	8000h	21.5.6/608
6FE	INTC Priority Select Register (INTC_PSR847)	16	R/W	8000h	21.5.6/608
700	INTC Priority Select Register (INTC_PSR848)	16	R/W	8000h	21.5.6/608
702	INTC Priority Select Register (INTC_PSR849)	16	R/W	8000h	21.5.6/608
704	INTC Priority Select Register (INTC_PSR850)	16	R/W	8000h	21.5.6/608
706	INTC Priority Select Register (INTC_PSR851)	16	R/W	8000h	21.5.6/608
708	INTC Priority Select Register (INTC_PSR852)	16	R/W	8000h	21.5.6/608
70A	INTC Priority Select Register (INTC_PSR853)	16	R/W	8000h	21.5.6/608
70C	INTC Priority Select Register (INTC_PSR854)	16	R/W	8000h	21.5.6/608
70E	INTC Priority Select Register (INTC_PSR855)	16	R/W	8000h	21.5.6/608
710	INTC Priority Select Register (INTC_PSR856)	16	R/W	8000h	21.5.6/608
712	INTC Priority Select Register (INTC_PSR857)	16	R/W	8000h	21.5.6/608
714	INTC Priority Select Register (INTC_PSR858)	16	R/W	8000h	21.5.6/608
716	INTC Priority Select Register (INTC_PSR859)	16	R/W	8000h	21.5.6/608
718	INTC Priority Select Register (INTC_PSR860)	16	R/W	8000h	21.5.6/608
71A	INTC Priority Select Register (INTC_PSR861)	16	R/W	8000h	21.5.6/608
71C	INTC Priority Select Register (INTC_PSR862)	16	R/W	8000h	21.5.6/608
71E	INTC Priority Select Register (INTC_PSR863)	16	R/W	8000h	21.5.6/608
720	INTC Priority Select Register (INTC_PSR864)	16	R/W	8000h	21.5.6/608
722	INTC Priority Select Register (INTC_PSR865)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
724	INTC Priority Select Register (INTC_PSR866)	16	R/W	8000h	21.5.6/608
726	INTC Priority Select Register (INTC_PSR867)	16	R/W	8000h	21.5.6/608
728	INTC Priority Select Register (INTC_PSR868)	16	R/W	8000h	21.5.6/608
72A	INTC Priority Select Register (INTC_PSR869)	16	R/W	8000h	21.5.6/608
72C	INTC Priority Select Register (INTC_PSR870)	16	R/W	8000h	21.5.6/608
72E	INTC Priority Select Register (INTC_PSR871)	16	R/W	8000h	21.5.6/608
730	INTC Priority Select Register (INTC_PSR872)	16	R/W	8000h	21.5.6/608
732	INTC Priority Select Register (INTC_PSR873)	16	R/W	8000h	21.5.6/608
734	INTC Priority Select Register (INTC_PSR874)	16	R/W	8000h	21.5.6/608
736	INTC Priority Select Register (INTC_PSR875)	16	R/W	8000h	21.5.6/608
738	INTC Priority Select Register (INTC_PSR876)	16	R/W	8000h	21.5.6/608
73A	INTC Priority Select Register (INTC_PSR877)	16	R/W	8000h	21.5.6/608
73C	INTC Priority Select Register (INTC_PSR878)	16	R/W	8000h	21.5.6/608
73E	INTC Priority Select Register (INTC_PSR879)	16	R/W	8000h	21.5.6/608
740	INTC Priority Select Register (INTC_PSR880)	16	R/W	8000h	21.5.6/608
742	INTC Priority Select Register (INTC_PSR881)	16	R/W	8000h	21.5.6/608
744	INTC Priority Select Register (INTC_PSR882)	16	R/W	8000h	21.5.6/608
746	INTC Priority Select Register (INTC_PSR883)	16	R/W	8000h	21.5.6/608
748	INTC Priority Select Register (INTC_PSR884)	16	R/W	8000h	21.5.6/608
74A	INTC Priority Select Register (INTC_PSR885)	16	R/W	8000h	21.5.6/608
74C	INTC Priority Select Register (INTC_PSR886)	16	R/W	8000h	21.5.6/608
74E	INTC Priority Select Register (INTC_PSR887)	16	R/W	8000h	21.5.6/608
750	INTC Priority Select Register (INTC_PSR888)	16	R/W	8000h	21.5.6/608
752	INTC Priority Select Register (INTC_PSR889)	16	R/W	8000h	21.5.6/608
754	INTC Priority Select Register (INTC_PSR890)	16	R/W	8000h	21.5.6/608
756	INTC Priority Select Register (INTC_PSR891)	16	R/W	8000h	21.5.6/608
758	INTC Priority Select Register (INTC_PSR892)	16	R/W	8000h	21.5.6/608
75A	INTC Priority Select Register (INTC_PSR893)	16	R/W	8000h	21.5.6/608
75C	INTC Priority Select Register (INTC_PSR894)	16	R/W	8000h	21.5.6/608
75E	INTC Priority Select Register (INTC_PSR895)	16	R/W	8000h	21.5.6/608
760	INTC Priority Select Register (INTC_PSR896)	16	R/W	8000h	21.5.6/608
762	INTC Priority Select Register (INTC_PSR897)	16	R/W	8000h	21.5.6/608
764	INTC Priority Select Register (INTC_PSR898)	16	R/W	8000h	21.5.6/608
766	INTC Priority Select Register (INTC_PSR899)	16	R/W	8000h	21.5.6/608
768	INTC Priority Select Register (INTC_PSR900)	16	R/W	8000h	21.5.6/608
76A	INTC Priority Select Register (INTC_PSR901)	16	R/W	8000h	21.5.6/608
76C	INTC Priority Select Register (INTC_PSR902)	16	R/W	8000h	21.5.6/608
76E	INTC Priority Select Register (INTC_PSR903)	16	R/W	8000h	21.5.6/608
770	INTC Priority Select Register (INTC_PSR904)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
772	INTC Priority Select Register (INTC_PSR905)	16	R/W	8000h	21.5.6/608
774	INTC Priority Select Register (INTC_PSR906)	16	R/W	8000h	21.5.6/608
776	INTC Priority Select Register (INTC_PSR907)	16	R/W	8000h	21.5.6/608
778	INTC Priority Select Register (INTC_PSR908)	16	R/W	8000h	21.5.6/608
77A	INTC Priority Select Register (INTC_PSR909)	16	R/W	8000h	21.5.6/608
77C	INTC Priority Select Register (INTC_PSR910)	16	R/W	8000h	21.5.6/608
77E	INTC Priority Select Register (INTC_PSR911)	16	R/W	8000h	21.5.6/608
780	INTC Priority Select Register (INTC_PSR912)	16	R/W	8000h	21.5.6/608
782	INTC Priority Select Register (INTC_PSR913)	16	R/W	8000h	21.5.6/608
784	INTC Priority Select Register (INTC_PSR914)	16	R/W	8000h	21.5.6/608
786	INTC Priority Select Register (INTC_PSR915)	16	R/W	8000h	21.5.6/608
788	INTC Priority Select Register (INTC_PSR916)	16	R/W	8000h	21.5.6/608
78A	INTC Priority Select Register (INTC_PSR917)	16	R/W	8000h	21.5.6/608
78C	INTC Priority Select Register (INTC_PSR918)	16	R/W	8000h	21.5.6/608
78E	INTC Priority Select Register (INTC_PSR919)	16	R/W	8000h	21.5.6/608
790	INTC Priority Select Register (INTC_PSR920)	16	R/W	8000h	21.5.6/608
792	INTC Priority Select Register (INTC_PSR921)	16	R/W	8000h	21.5.6/608
794	INTC Priority Select Register (INTC_PSR922)	16	R/W	8000h	21.5.6/608
796	INTC Priority Select Register (INTC_PSR923)	16	R/W	8000h	21.5.6/608
798	INTC Priority Select Register (INTC_PSR924)	16	R/W	8000h	21.5.6/608
79A	INTC Priority Select Register (INTC_PSR925)	16	R/W	8000h	21.5.6/608
79C	INTC Priority Select Register (INTC_PSR926)	16	R/W	8000h	21.5.6/608
79E	INTC Priority Select Register (INTC_PSR927)	16	R/W	8000h	21.5.6/608
7A0	INTC Priority Select Register (INTC_PSR928)	16	R/W	8000h	21.5.6/608
7A2	INTC Priority Select Register (INTC_PSR929)	16	R/W	8000h	21.5.6/608
7A4	INTC Priority Select Register (INTC_PSR930)	16	R/W	8000h	21.5.6/608
7A6	INTC Priority Select Register (INTC_PSR931)	16	R/W	8000h	21.5.6/608
7A8	INTC Priority Select Register (INTC_PSR932)	16	R/W	8000h	21.5.6/608
7AA	INTC Priority Select Register (INTC_PSR933)	16	R/W	8000h	21.5.6/608
7AC	INTC Priority Select Register (INTC_PSR934)	16	R/W	8000h	21.5.6/608
7AE	INTC Priority Select Register (INTC_PSR935)	16	R/W	8000h	21.5.6/608
7B0	INTC Priority Select Register (INTC_PSR936)	16	R/W	8000h	21.5.6/608
7B2	INTC Priority Select Register (INTC_PSR937)	16	R/W	8000h	21.5.6/608
7B4	INTC Priority Select Register (INTC_PSR938)	16	R/W	8000h	21.5.6/608
7B6	INTC Priority Select Register (INTC_PSR939)	16	R/W	8000h	21.5.6/608
7B8	INTC Priority Select Register (INTC_PSR940)	16	R/W	8000h	21.5.6/608
7BA	INTC Priority Select Register (INTC_PSR941)	16	R/W	8000h	21.5.6/608
7BC	INTC Priority Select Register (INTC_PSR942)	16	R/W	8000h	21.5.6/608
7BE	INTC Priority Select Register (INTC_PSR943)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7C0	INTC Priority Select Register (INTC_PSR944)	16	R/W	8000h	21.5.6/608
7C2	INTC Priority Select Register (INTC_PSR945)	16	R/W	8000h	21.5.6/608
7C4	INTC Priority Select Register (INTC_PSR946)	16	R/W	8000h	21.5.6/608
7C6	INTC Priority Select Register (INTC_PSR947)	16	R/W	8000h	21.5.6/608
7C8	INTC Priority Select Register (INTC_PSR948)	16	R/W	8000h	21.5.6/608
7CA	INTC Priority Select Register (INTC_PSR949)	16	R/W	8000h	21.5.6/608
7CC	INTC Priority Select Register (INTC_PSR950)	16	R/W	8000h	21.5.6/608
7CE	INTC Priority Select Register (INTC_PSR951)	16	R/W	8000h	21.5.6/608
7D0	INTC Priority Select Register (INTC_PSR952)	16	R/W	8000h	21.5.6/608
7D2	INTC Priority Select Register (INTC_PSR953)	16	R/W	8000h	21.5.6/608
7D4	INTC Priority Select Register (INTC_PSR954)	16	R/W	8000h	21.5.6/608
7D6	INTC Priority Select Register (INTC_PSR955)	16	R/W	8000h	21.5.6/608
7D8	INTC Priority Select Register (INTC_PSR956)	16	R/W	8000h	21.5.6/608
7DA	INTC Priority Select Register (INTC_PSR957)	16	R/W	8000h	21.5.6/608
7DC	INTC Priority Select Register (INTC_PSR958)	16	R/W	8000h	21.5.6/608
7DE	INTC Priority Select Register (INTC_PSR959)	16	R/W	8000h	21.5.6/608
7E0	INTC Priority Select Register (INTC_PSR960)	16	R/W	8000h	21.5.6/608
7E2	INTC Priority Select Register (INTC_PSR961)	16	R/W	8000h	21.5.6/608
7E4	INTC Priority Select Register (INTC_PSR962)	16	R/W	8000h	21.5.6/608
7E6	INTC Priority Select Register (INTC_PSR963)	16	R/W	8000h	21.5.6/608
7E8	INTC Priority Select Register (INTC_PSR964)	16	R/W	8000h	21.5.6/608
7EA	INTC Priority Select Register (INTC_PSR965)	16	R/W	8000h	21.5.6/608
7EC	INTC Priority Select Register (INTC_PSR966)	16	R/W	8000h	21.5.6/608
7EE	INTC Priority Select Register (INTC_PSR967)	16	R/W	8000h	21.5.6/608
7F0	INTC Priority Select Register (INTC_PSR968)	16	R/W	8000h	21.5.6/608
7F2	INTC Priority Select Register (INTC_PSR969)	16	R/W	8000h	21.5.6/608
7F4	INTC Priority Select Register (INTC_PSR970)	16	R/W	8000h	21.5.6/608
7F6	INTC Priority Select Register (INTC_PSR971)	16	R/W	8000h	21.5.6/608
7F8	INTC Priority Select Register (INTC_PSR972)	16	R/W	8000h	21.5.6/608
7FA	INTC Priority Select Register (INTC_PSR973)	16	R/W	8000h	21.5.6/608
7FC	INTC Priority Select Register (INTC_PSR974)	16	R/W	8000h	21.5.6/608
7FE	INTC Priority Select Register (INTC_PSR975)	16	R/W	8000h	21.5.6/608
800	INTC Priority Select Register (INTC_PSR976)	16	R/W	8000h	21.5.6/608
802	INTC Priority Select Register (INTC_PSR977)	16	R/W	8000h	21.5.6/608
804	INTC Priority Select Register (INTC_PSR978)	16	R/W	8000h	21.5.6/608
806	INTC Priority Select Register (INTC_PSR979)	16	R/W	8000h	21.5.6/608
808	INTC Priority Select Register (INTC_PSR980)	16	R/W	8000h	21.5.6/608
80A	INTC Priority Select Register (INTC_PSR981)	16	R/W	8000h	21.5.6/608
80C	INTC Priority Select Register (INTC_PSR982)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80E	INTC Priority Select Register (INTC_PSR983)	16	R/W	8000h	21.5.6/608
810	INTC Priority Select Register (INTC_PSR984)	16	R/W	8000h	21.5.6/608
812	INTC Priority Select Register (INTC_PSR985)	16	R/W	8000h	21.5.6/608
814	INTC Priority Select Register (INTC_PSR986)	16	R/W	8000h	21.5.6/608
816	INTC Priority Select Register (INTC_PSR987)	16	R/W	8000h	21.5.6/608
818	INTC Priority Select Register (INTC_PSR988)	16	R/W	8000h	21.5.6/608
81A	INTC Priority Select Register (INTC_PSR989)	16	R/W	8000h	21.5.6/608
81C	INTC Priority Select Register (INTC_PSR990)	16	R/W	8000h	21.5.6/608
81E	INTC Priority Select Register (INTC_PSR991)	16	R/W	8000h	21.5.6/608
820	INTC Priority Select Register (INTC_PSR992)	16	R/W	8000h	21.5.6/608
822	INTC Priority Select Register (INTC_PSR993)	16	R/W	8000h	21.5.6/608
824	INTC Priority Select Register (INTC_PSR994)	16	R/W	8000h	21.5.6/608
826	INTC Priority Select Register (INTC_PSR995)	16	R/W	8000h	21.5.6/608
828	INTC Priority Select Register (INTC_PSR996)	16	R/W	8000h	21.5.6/608
82A	INTC Priority Select Register (INTC_PSR997)	16	R/W	8000h	21.5.6/608
82C	INTC Priority Select Register (INTC_PSR998)	16	R/W	8000h	21.5.6/608
82E	INTC Priority Select Register (INTC_PSR999)	16	R/W	8000h	21.5.6/608
830	INTC Priority Select Register (INTC_PSR1000)	16	R/W	8000h	21.5.6/608
832	INTC Priority Select Register (INTC_PSR1001)	16	R/W	8000h	21.5.6/608
834	INTC Priority Select Register (INTC_PSR1002)	16	R/W	8000h	21.5.6/608
836	INTC Priority Select Register (INTC_PSR1003)	16	R/W	8000h	21.5.6/608
838	INTC Priority Select Register (INTC_PSR1004)	16	R/W	8000h	21.5.6/608
83A	INTC Priority Select Register (INTC_PSR1005)	16	R/W	8000h	21.5.6/608
83C	INTC Priority Select Register (INTC_PSR1006)	16	R/W	8000h	21.5.6/608
83E	INTC Priority Select Register (INTC_PSR1007)	16	R/W	8000h	21.5.6/608
840	INTC Priority Select Register (INTC_PSR1008)	16	R/W	8000h	21.5.6/608
842	INTC Priority Select Register (INTC_PSR1009)	16	R/W	8000h	21.5.6/608
844	INTC Priority Select Register (INTC_PSR1010)	16	R/W	8000h	21.5.6/608
846	INTC Priority Select Register (INTC_PSR1011)	16	R/W	8000h	21.5.6/608
848	INTC Priority Select Register (INTC_PSR1012)	16	R/W	8000h	21.5.6/608
84A	INTC Priority Select Register (INTC_PSR1013)	16	R/W	8000h	21.5.6/608
84C	INTC Priority Select Register (INTC_PSR1014)	16	R/W	8000h	21.5.6/608
84E	INTC Priority Select Register (INTC_PSR1015)	16	R/W	8000h	21.5.6/608
850	INTC Priority Select Register (INTC_PSR1016)	16	R/W	8000h	21.5.6/608
852	INTC Priority Select Register (INTC_PSR1017)	16	R/W	8000h	21.5.6/608
854	INTC Priority Select Register (INTC_PSR1018)	16	R/W	8000h	21.5.6/608
856	INTC Priority Select Register (INTC_PSR1019)	16	R/W	8000h	21.5.6/608
858	INTC Priority Select Register (INTC_PSR1020)	16	R/W	8000h	21.5.6/608
85A	INTC Priority Select Register (INTC_PSR1021)	16	R/W	8000h	21.5.6/608

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
85C	INTC Priority Select Register (INTC_PSR1022)	16	R/W	8000h	21.5.6/608
85E	INTC Priority Select Register (INTC_PSR1023)	16	R/W	8000h	21.5.6/608
1000	INTC Monitor Mode Register 0 (INTC_MMRC0)	32	R/W	0000_0000h	21.5.7/609
1004	INTC HIPRI Register (INTC_HIPRI0C0)	32	R/W	0000_0000h	21.5.8/610
1008	INTC HIPRI Register (INTC_HIPRI1C0)	32	R/W	0000_0000h	21.5.8/610
100C	INTC HIPRI Register (INTC_HIPRI2C0)	32	R/W	0000_0000h	21.5.8/610
1014	INTC LAT Register (INTC_LAT0C0)	32	R/W	0000_0000h	21.5.9/611
1018	INTC LAT Register (INTC_LAT1C0)	32	R/W	0000_0000h	21.5.9/611
101C	INTC LAT Register (INTC_LAT2C0)	32	R/W	0000_0000h	21.5.9/611
1024	INTC Timer Register (INTC_TIMER0C0)	32	R/W	0000_0000h	21.5.10/ 611
1028	INTC Timer Register (INTC_TIMER1C0)	32	R/W	0000_0000h	21.5.10/ 611
102C	INTC Timer Register (INTC_TIMER2C0)	32	R/W	0000_0000h	21.5.10/ 611

21.5.1 INTC Block Configuration Register (INTC_BCR)

The Block Configuration Register is used to configure options of the INTC.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			0	0			0	0			0	0			HVEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_BCR field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

INTC_BCR field descriptions (continued)

Field	Description
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVEN0	Hardware vector enable for processor 0. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

21.5.2 INTC Current Priority Register for Processor 0 (INTC_CPR0)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											PRI				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

INTC_CPR0 field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PRI	Priority of the currently executing ISR, according to the field values 31 (highest priority) down to 0 (lowest priority).

21.5.3 INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VTBA																				INTVEC										0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_IACKR0 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.5.4 INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)

Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	EOI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_EOIR0 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

21.5.5 INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET will leave SET unchanged at 0 but will set CLR. Writing a 0 to SET will have no effect. CLR is the flag bit. Writing a 1 to CLR will clear it. Writing a 0 to CLR will have no effect. If a 1 is written to a pair of SET and CLR bits at the same time, CLR is asserted, regardless of whether CLR was asserted before the write.

See Figure 21-2 for limitations on writing to this register.

Address: 0h base + 40h offset + (1d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7
Read	0						0	CLR
Write							SET	w1c
Reset	0	0	0	0	0	0	0	0

INTC_SSCIR*n* field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SET	Set flag bits. Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
7 CLR	Clear flag bits. CLR is the flag bit. Writing a 1 to CLR will clear it provided that a 1 is not written simultaneously to its corresponding SET bit. Writing a 0 to CLR will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

21.5.6 INTC Priority Select Register (INTC_PSR*n*)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSRn is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32-1023, and their priorities are configured in INTC_PSR32-1023, respectively.

See Figure 21-2 for limitations on writing to this register.

NOTE

Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.

NOTE

Interrupts are not detected while the PSRs are being written.

Address: 0h base + 60h offset + (2d × i), where i=0d to 1023d

Bit	0	1	2	3	4	5	6	7
Read	PRC_ SELN0	0	0	0	0			SWTN
Write								
Reset	1	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			PRIN				
Write								
Reset	0	0	0	0	0	0	0	0

INTC_PSR*n* field descriptions

Field	Description
0 PRC_SELN0	Processor select 0. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 0. 0 Interrupt request not sent to processor 0 1 Interrupt request sent to processor 0
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SWTN	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a '1' to SWT*n* sets the bit. On the interrupt acknowledge cycle, the SWT*n* bit is cleared. The SWT function works both in hardware and software vector mode. It is only cleared by the enabled Processor(s) defined in the PRC_SEL*n* field of the PSR.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PRIN	Priority select. Selects the priority for the interrupt requests. PRI has values 31 (highest priority) down to 0 (lowest priority).

21.5.7 INTC Monitor Mode Register 0 (INTC_MMRC0)

The Monitor Mode Register is used to configure measurement options of the INTC.

Address: 0h base + 1000h offset = 1000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														MM	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_MMRC0 field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 MM	Monitor Mode. Controls whether the INTC monitors the latency or interrupt handler time 00 Disabled 01 IACK - Interrupt Acknowledge - timer is active from assertion of IRQ until interrupt acknowledge. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 10 EOI - End Of Interrupt - timer is active from assertion of IRQ until the end of the interrupt handler. The timer will continue to run even if interrupt handler is preempted. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 11 SW - Software - timer is active from assertion of IRQ until the end of the interrupt handler unless it is preempted. A return from interrupt will start the timer if it is the timer associated with the active IRQ. No error is signaled; rather software can read the timer at any time.

21.5.8 INTC HIPRI Register (INTC_HIPRI*n*C0)

The HIPRI log2(SOURCES)-bit registers are used to indicate which IRQ is to be monitored or checked. The mnemonic is of the form HIPRImCn, where the 'n' is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1004h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						IRQ									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_HIPRI*n*C0 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 IRQ	Select the IRQ to monitor (input the hex value of the IRQ)

21.5.9 INTC LAT Register (INTC_LAT*n*C0)

These registers are used to indicate the maximum time before an error signal is asserted for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form LATmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1014h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LAT																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_LAT*n*C0 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LAT	Select the maximum number of INTC clock cycles allowed for the monitored IRQ.

21.5.10 INTC Timer Register (INTC_TIMER*n*C0)

These registers are used to indicate the time for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form TIMERmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor are used to indicate the maximum time for a detected IRQ to either IACK or RFI based on the Mode.

Address: 0h base + 1024h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TIMER																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_TIMER*n*C0 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TIMER	Timer is used to count the number of INTC clock cycles up to 24 bits of resolution.

21.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

21.6.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI*n* value in INTC_PSR*n* is higher than the PRI value in INTC_CPR*n*, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in the associated INTC_CPR*n* is updated with the corresponding PRI*n* value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

21.6.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

21.6.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR*n*) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET bit in INTC_SSCIR*n*. This write sets the corresponding CLR bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR bit. Specific behavior includes the following:

- Writing a 1 to SET leaves SET unchanged at 0 but sets the flag bit (which is the CLR bit).
- Writing a 0 to SET has no effect.
- Writing a 1 to CLR clears the flag (CLR) bit.
- Writing a 0 to CLR has no effect.
- If a 1 is written to a pair of SET and CLR bits at the same time, the flag (CLR) is set, regardless of whether CLR was asserted before the write.

The time from the write to the SET bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

Any Processor n will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the Processor n which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSRn[PRC_SELn] bits.

21.6.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–15 are assigned vectors 0–15, respectively. The peripheral interrupt requests are assigned vectors from 16 to a number as high as needed to cover all peripheral interrupt requests.

21.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL*n* values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in the associated INTC_CPR*n*. The results of those comparisons are used to manage the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

21.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in Figure 21-1 are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for the associated INTC_IACKR*n*, and if in hardware vector mode, for the interrupt vector provided to the processor.

21.6.2.1.1 Priority tree

The priority tree for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated selector logic.

21.6.2.1.2 Selector

If only one interrupt request from the associated priority tree is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority tree are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

21.6.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

21.6.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated INTC_CPR*n*. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR*n*, or the PRI value in the associated INTC_CPR*n* is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated

INTC_CPR*n* when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in the associated INTC_CPR*n*.

Another function of the comparator is to signal an update of the INTC_IACKR*n* with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of the INTC_CPR*n* occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

21.6.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated INTC_CPR*n*. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the associated INTC_CPR*n* does not need to be loaded from the associated INTC_CPR*n* and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the associated INTC_CPR*n*.

The PRI value in the associated INTC_CPR*n* is pushed onto the LIFO when the associated INTC_IACKR*n* is read in software vector mode or the interrupt acknowledge signal from the associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated INTC_CPR*n* whenever the associated INTC_EOIR*n* is written. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

Although the INTC supports up to 32 priorities, an ISR executing with PRI in the INTC_CPR*n* equal to 31 will not be preempted. Therefore, the LIFO supports the stacking of 31 priorities. However, the LIFO is only 30 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 31 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

21.6.3 Handshaking with processor

This section describes handshaking with the processor.

21.6.3.1 Software vector mode handshaking

This section describes the software vector mode handshaking.

21.6.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to Figure 21-3.

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 21-3. The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in Software vector mode.

21.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR*n* must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated INTC_CPR*n*. Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in the associated INTC_CPR*n* is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

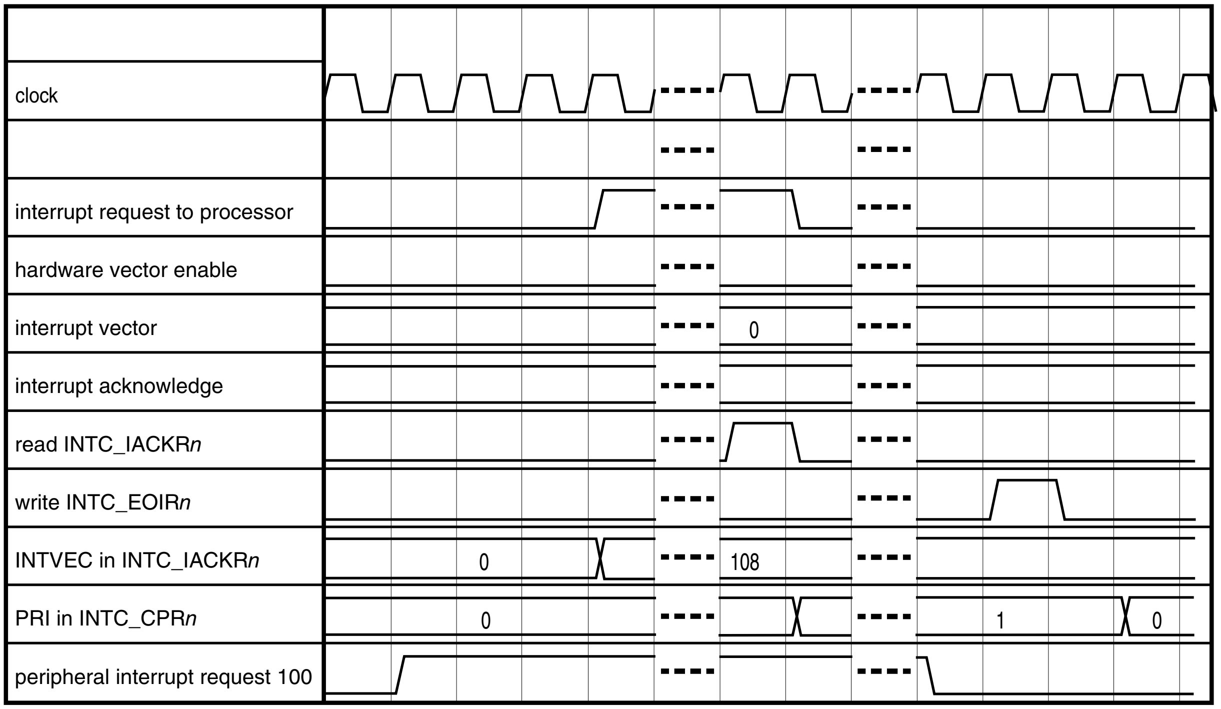


Figure 21-3. Timing diagram of software vector mode handshaking

21.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 21-4. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKR*n*. The rest of the handshaking is described in Hardware vector mode.

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIR*n*, is the same as in software vector mode. See End of interrupt exception handler.

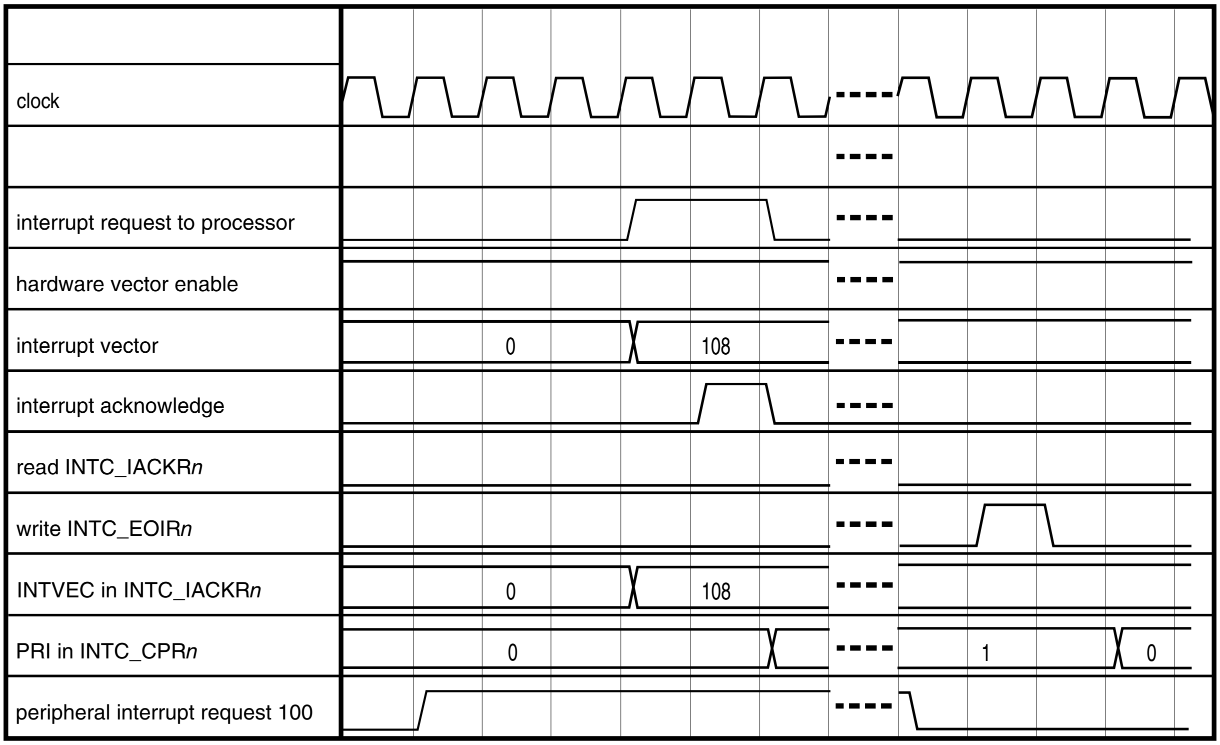


Figure 21-4. Timing diagram for hardware vector mode handshaking

21.6.4 Interrupt Controller Monitor (INTCM)

Isolated in its own hierarchy, the interrupt latency monitor logic is used to monitor or check interrupt latency for reliability, safety, and interrupt software requirements.

21.6.4.1 Operation

The following section lists the operation of the INTCM.

21.6.4.1.1 TIMERn

A clear of TIMERn will clear the corresponding error.

The TIMERn will not run if there is a corresponding error.

If the INTC_MMRn[MM] is enabled and an error is signaled, disable TIMERn.

If the INTC_MMRn[MM] is enabled, for a transition from 0 to 1 of SOURCE input that is being monitored by INTC_HIPRIn registers, clear and enable TIMERn. Once the timer reaches the maximum count, it will stop and not roll over. Software can read the TIMERn value at any time.

If the INTC_MMRn[MM] is enabled and if the TIMERn is enabled, then it should increment each cycle.

If the INTC_MMRn[MM] is enabled and if INTC_LATn is non-zero, an error will be signaled if the timer exceeds INTC_LATn or if the TIMERn reaches the maximum count.

If the INTC_MMRn[MM] is IACK and an interrupt acknowledge asserts for the monitored source, INTC_HIPRIn[IRQ], then the enabled TIMERn should stop. As an extra check, the IRQ is used to ensure that the acknowledged interrupt matches the monitored interrupt INTC_HIPRIn[IRQ].

If the INTC_MMRn[MM] is SW and an interrupt acknowledge occurs where a monitored interrupt is preempted, the enabled TIMERn should stop.

If the INTC_MMRn[MM] is EOI or SW for a monitored interrupt which has an INTC_EOIRn, the enabled TIMERn should stop.

If the INTC_MMRn[MM] is SW and if INTC_EOIRn, and if the previous one is monitored, it will be re-enabled.

21.7 Initialization/application information

This section contains initialization/application information.

21.7.1 Initialization flow

After exiting reset, all of the PRI*n* and PRC_SEL*n* fields in INTC_PSR*n* are zero, and PRI in the INTC_CPRn registers is 31. These reset values will prevent the INTC from asserting the interrupt request to the processors. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- interrupt_request_initialization:
    - Configure HVENn in INTC_BCR.
    - Configure VTBAn in INTC_IACKRn.
    - Raise the PRIn fields and set the PRC_SELn fields to the desired processor in INTC_PSRn.
    - Set the enable bits or clear the mask bits for the peripheral interrupt requests.
    - Lower PRI in INTC_CPRn to zero.
    - Enable processor(s) recognition of interrupts.

21.7.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

21.7.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 bytes of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
b        interrupt_exception_handler_continued# 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1 (Power
Architecture Save/Restore registers) (not shown)

lis        r3,INTC_IACKRn@ha        # form adjusted upper half of INTC_IACKRn address
lwz       r3,INTC_IACKRn@l(r3)      # load INTC_IACKRn, which clears request to processor
lwz       r3,0x0(r3)                 # load address of ISR from vector table
wrteei    1                        # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded Binary
Application Interface) (not shown)

mtlr    r3                        # move the INTC_IACKRn address into the link register
blrl                            # branch to ISR; link register updated with epilog
                                # address
epilog:
code to restore most of context required by Power Architecture EABI (not shown)
# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                            # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                    # form 0 to write to INTC_EOIRn
wrteei    0                        # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)        # store to INTC_EOIRn, informing INTC to lower priority
code to restore SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
        .
        .
        .
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023
ISRn:
code to service the interrupt event (not shown)
code to clear flag bit which drives interrupt request to INTC (not shown)
blr                                # return to epilog
```

21.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b        interrupt_exception_handler_continuedn# 16 bytes available, branch to continue

interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1 (not shown)

wrteei    1                    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)

bl        ISRn                # branch to ISR for interrupt with vector n

epilog:
code to restore most of context required by Power Architecture EABI (not shown)

# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                            # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                # form 0 to write to INTC_EOIRn
wrteei    0                    # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)    # store to INTC_EOIRn, informing INTC to lower priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr        # branch to epilog
```

21.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR*n* having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR*n* priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR*n* priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPRn priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR*n* while the shared resource is being accessed.

An ISR whose PRI*n* in INTC_PSR*n* has a value of 0 will not cause an interrupt request to the selected processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

21.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in the following table shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 21-1. Order of ISR execution example

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3

Table continues on the next page...

Table 21-1. Order of ISR execution example (continued)

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108[1]	ISR208	ISR308	ISR408	interrupt exception handler	
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	0
12	RTOS continues execution.	X						0

[1] ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

21.7.5 Priority ceiling protocol

This section describes the priority ceiling protocol.

21.7.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR*n*) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR*n* to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR*n* to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

21.7.5.2 Ensuring coherency

This section discusses how to ensure coherency.

21.7.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR*n*. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR*n* can be made by those system services with the code sequence:

```
GetResource:
        wrteei 0                # disable external interrupts to the Processor
        raise PRI                # Write to CPR, cache inhibit, guarded
        mbar                    # flush out writes from store buffer
        wrteei 1                # enable external interrupts to the Processor
        isync                    # re-fetch Processor pipeline

ReleaseResource:
        mbar                    # flush out writes from store buffer
        wrteei 0                # disable external interrupts to the Processor
        lower PRI                # Write to CPR, cache inhibit, guarded
        wrteei 1                # enable external interrupts to the Processor
```

21.7.5.2.1.1 Interrupt request to processor

Referencing Figure 21-1, the interrupt request logic to the processor is shown in the following figure.

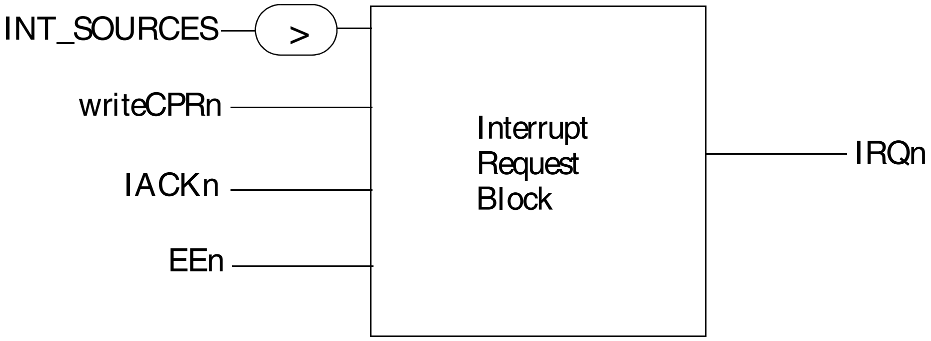


Figure 21-5. Interrupt Request Block Diagram

Assuming that there are no IRQn deasserted (no pending interrupt requests to processor n), if one of the external interrupt sources (INT_SOURCES) has a priority higher than the INTC_CPRn, then interrupt request to the processor (IRQn) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge (IACK) is asserted
- If the EEn bit has been cleared by the wrteei instruction (see the code sequence in Interrupt with blocked priority), IRQn will be re-evaluated when processor n writes to the INTC_CPRn while processor n ignores IRQn

This provides a safe way to guarantee that no interrupts will be recognized by processor n while updating the INTC_CPRn.

21.7.5.2.2 Raised priority preserved

Before the instruction after the GetResource system service executes, all pending transactions have completed. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. The shared coherent data block now can be accessed coherently. The following figure shows the timing diagram for this scenario, and the following table explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

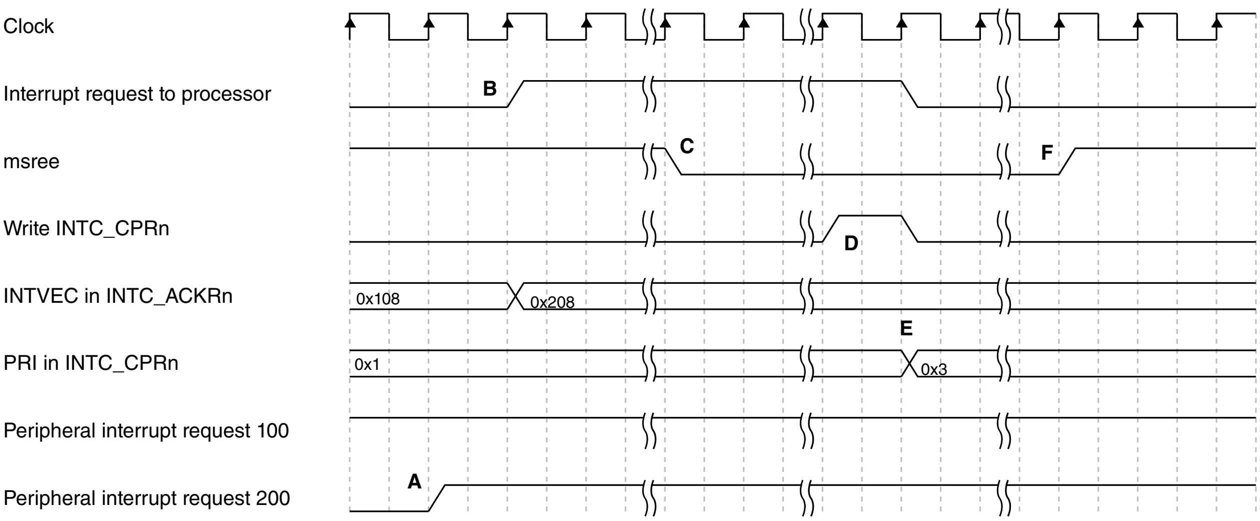

Clock
Interrupt request to processor
msree
Write INTC_CPRn
INTVEC in INTC_ACKRn
PRI in INTC_CPRn
Peripheral interrupt request 100
Peripheral interrupt request 200
A
B
C
D
E
F
0x108
0x208
0x1
0x3

Figure 21-6. Timing diagram of raised priority preserved

Table 21-2. Raised priority preserved events

Event	Description
A	Peripheral interrupt request 200 of priority 2 asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKRn updates with vector for that peripheral interrupt request.
C	Write to msree bit to disable external interrupts
D	ISR108 writes to INTC_CPRn to raise priority to 3 before accessing shared coherent data block.
E	PRI in INTC_CPRn now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
F	Write to msree bit to enable external interrupts

21.7.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 32 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

21.7.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

21.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRIn value in INTC_PSR*n*, which becomes the PRI value in INTC_CPR*n* with the interrupt acknowledge. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET bit in INTC_SSCIR*n*. Writing a '1' to SET causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI*n* value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

21.7.7.2 Scheduling an ISR on another processor

Since the SET bits in the INTC_SSCIR*n* are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR bit in INTC_SSCIR*n* is asserted before again writing a 1 to the SET bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR bit and then writes 1 to a SET bit on the first processor, informing it that it now can access the block of data.

21.7.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR*n*, one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in Scheduling a lower priority portion of an ISR) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note

> Lowering the PRI value in INTC_CPR*n* within an ISR to below the ISR's corresponding PRI value in INTC_PSR*n* allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

21.7.9 Negating an interrupt request outside of its ISR

This section discusses the negation of an interrupt service request outside of its ISR.

21.7.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

21.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

21.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI*n* values in INTC_PSR*n* must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR*n* as the clearing of the flag bit that caused the present ISR to be executed. Refer to End of interrupt exception handler for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI*n* value in INTC_PSR*n*.

21.7.10 Examining LIFO contents

There are multiple methods for tracking interrupts in a system, one of which is described here using existing hardware (LIFO) within the INTC. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR*n*). To avoid a lower-level interrupt being serviced because the popping of the LIFO updates the INTC_CPR*n*, the processor recognition of interrupts should be disabled when examining LIFO contents. The pseudo-code is as follows:

```
      wrteei 0                # disable processor recognition of external interrupts
      oldCPRn = INTC_CPRn;    # save INTC_CPRn
      (branch to the pop_lifo)

pop_lifo:
      store INTC_EIORn        # pop INTC_CPR from LIFO, examine PRI, etc...
      examine INTC_CPRn[PRI], and store onto stack
      if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
      branch to push_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
      load stacked PRI value and store to INTC_CPRn
      load INTC_IACKn         # IACK; push INTC_CPRn into LIFO
      if stacked PRI values are not depleted, branch to push_lifo

      INTC_CPRn    = oldCPRn; # restore original INTC_CPRn and re-evaluate pending external
interrupts
      wrteei 1                # enable processor recognition of interrupts
```

NOTE

Disabling the processor recognition of interrupts during LIFO examination will introduce latency, but none of the interrupt requests will be missed.

21.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chip-specific INTC information.

Chapter 22
Enhanced Direct Memory Access (eDMA)

22.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 32 channels

22.1.1 eDMA system block diagram

Figure 22-1 illustrates the components of the eDMA system, including the eDMA module ("engine").

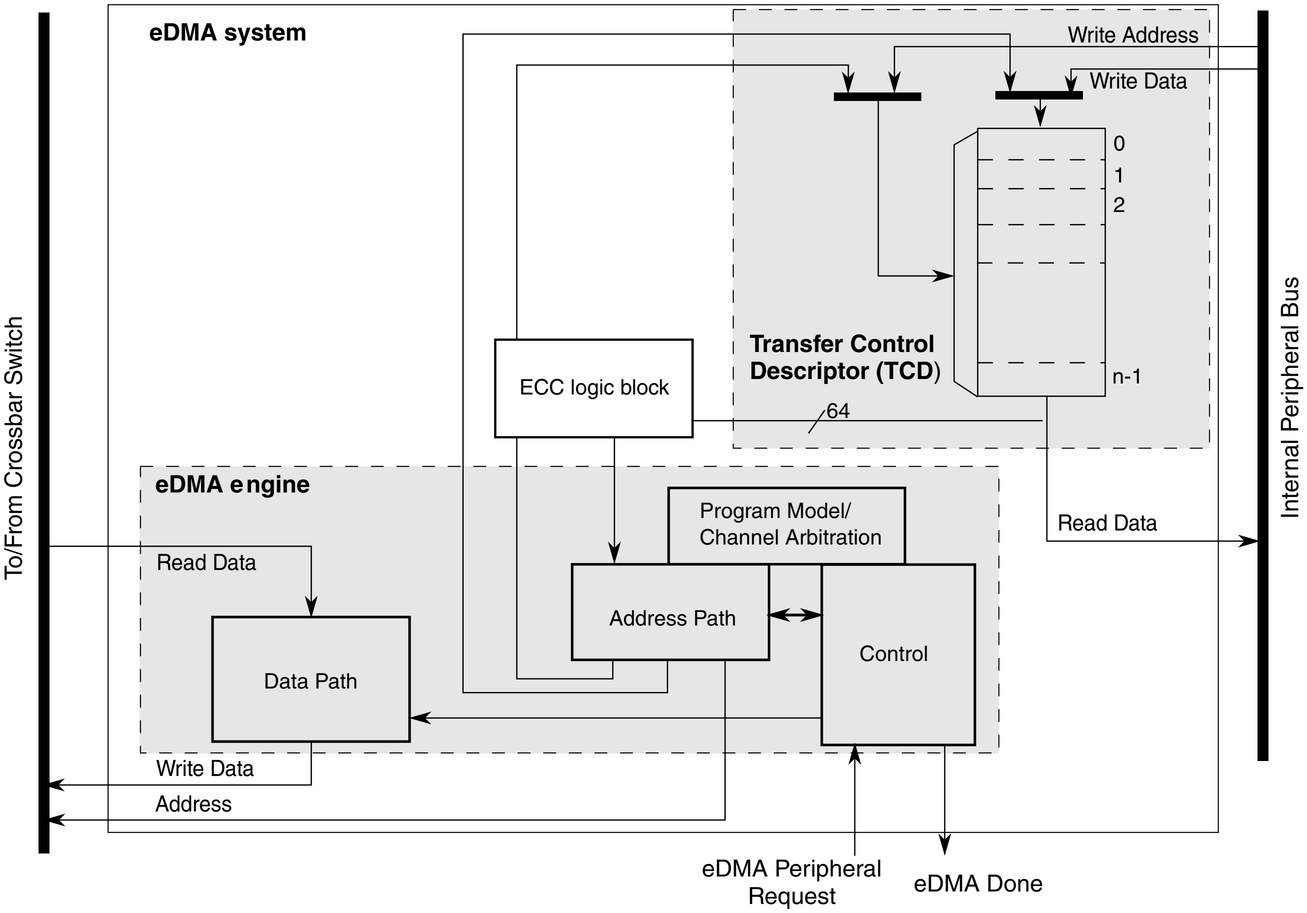


Figure 22-1. eDMA system block diagram

22.1.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

Table 22-1. eDMA engine submodules

Submodule	Function
Address path	This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes

Table continues on the next page...

Table 22-1. eDMA engine submodules (continued)

Submodule	Function
	the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	This block implements the bus master read/write datapath. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

The transfer-control descriptor local memory is further partitioned into:

Table 22-2. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
Memory array	TCD storage for each channel's transfer profile.

22.1.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
    - Programmable source and destination addresses and transfer size
    - Support for enhanced addressing modes

- 32-channel implementation that performs complex data transfers with minimal intervention from a host processor
    - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
    - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
    - 32-byte TCD stored in local memory for each channel
    - An inner data transfer loop defined by a minor byte transfer count
    - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
    - Explicit software initiation
    - Initiation via a channel-to-channel linking mechanism for continuous transfers
    - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
    - One interrupt per channel, which can be asserted at completion of major iteration count
    - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures
- Error detection and error correction

In the discussion of this module, *n* is used to reference the channel number.

22.2 Modes of operation

The eDMA operates in the following modes:

Table 22-3. Modes of operation

Mode	Description
Normal	In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA. A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.
Debug	DMA operation is configurable in Debug mode via the control register: • If CR[EDBG] is cleared, the DMA continues to operate. • If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.
Wait	Before entering Wait mode, the DMA attempts to complete its current transfer. After the transfer completes, the device enters Wait mode.

22.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

22.3.1 TCD memory

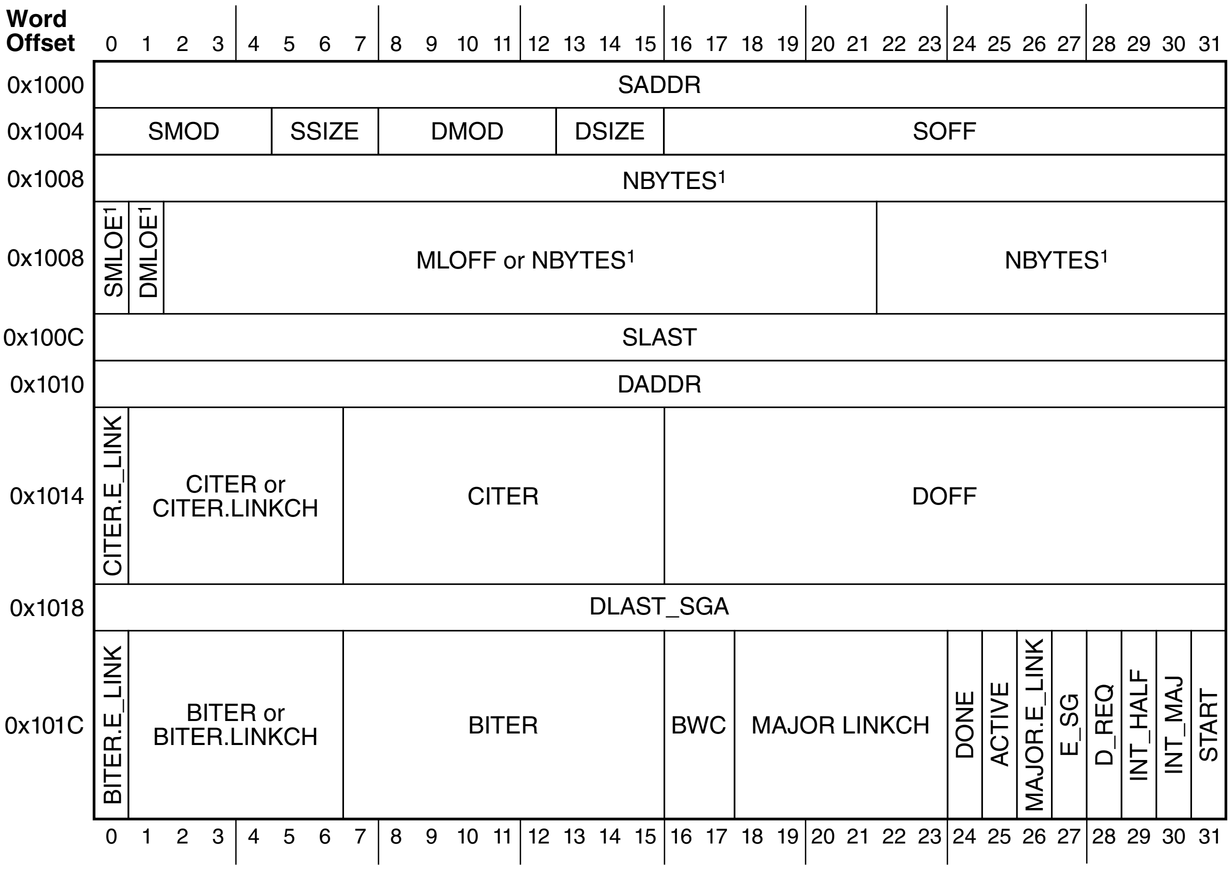
Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 31. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

22.3.2 TCD initialization

The initialization of the TCD memory is performed by the eDMA controller after reset. The initialization runs for 256 eDMA clock cycles. All fields in the TCD memory are initialized to 0. All application read or write accesses to the TCD are delayed until the initialization is finished. Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

22.3.3 TCD structure

DMA Basics: TCD Structure
- One DMA engine has a number of channels to react to DMA requests
- Each channel has its own TCD



[1] The fields implemented in Word 2 depend on whether EDMA_CR[EMLM] bit is set to 0 or 1.

22.3.4 Reserved memory and bit fields

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

DMA memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (DMA_CR)	32	R/W	0000_0400h	22.3.1/663

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Error Status Register (DMA_ES)	32	R	0000_0000h	22.3.2/666
C	Enable Request Register (DMA_ERQ)	32	R/W	0000_0000h	22.3.3/669
14	Enable Error Interrupt Register (DMA_EEI)	32	R/W	0000_0000h	22.3.4/672
18	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	22.3.5/676
19	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	22.3.6/677
1A	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	22.3.7/678
1B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	22.3.8/679
1C	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	22.3.9/680
1D	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	22.3.10/ 681
1E	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	22.3.11/ 682
1F	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	22.3.12/ 683
24	Interrupt Request Register (DMA_INT)	32	R/W	0000_0000h	22.3.13/ 684
2C	Error Register (DMA_ERR)	32	R/W	0000_0000h	22.3.14/ 687
34	Hardware Request Status Register (DMA_HRS)	32	R	0000_0000h	22.3.15/ 691
100	Channel n Priority Register (DMA_DCHPRI0)	8	R/W	See section	22.3.16/ 697
101	Channel n Priority Register (DMA_DCHPRI1)	8	R/W	See section	22.3.16/ 697
102	Channel n Priority Register (DMA_DCHPRI2)	8	R/W	See section	22.3.16/ 697
103	Channel n Priority Register (DMA_DCHPRI3)	8	R/W	See section	22.3.16/ 697
104	Channel n Priority Register (DMA_DCHPRI4)	8	R/W	See section	22.3.16/ 697
105	Channel n Priority Register (DMA_DCHPRI5)	8	R/W	See section	22.3.16/ 697

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
106	Channel n Priority Register (DMA_DCHPRI6)	8	R/W	See section	22.3.16/ 697
107	Channel n Priority Register (DMA_DCHPRI7)	8	R/W	See section	22.3.16/ 697
108	Channel n Priority Register (DMA_DCHPRI8)	8	R/W	See section	22.3.16/ 697
109	Channel n Priority Register (DMA_DCHPRI9)	8	R/W	See section	22.3.16/ 697
10A	Channel n Priority Register (DMA_DCHPRI10)	8	R/W	See section	22.3.16/ 697
10B	Channel n Priority Register (DMA_DCHPRI11)	8	R/W	See section	22.3.16/ 697
10C	Channel n Priority Register (DMA_DCHPRI12)	8	R/W	See section	22.3.16/ 697
10D	Channel n Priority Register (DMA_DCHPRI13)	8	R/W	See section	22.3.16/ 697
10E	Channel n Priority Register (DMA_DCHPRI14)	8	R/W	See section	22.3.16/ 697
10F	Channel n Priority Register (DMA_DCHPRI15)	8	R/W	See section	22.3.16/ 697
110	Channel n Priority Register (DMA_DCHPRI16)	8	R/W	See section	22.3.16/ 697
111	Channel n Priority Register (DMA_DCHPRI17)	8	R/W	See section	22.3.16/ 697
112	Channel n Priority Register (DMA_DCHPRI18)	8	R/W	See section	22.3.16/ 697
113	Channel n Priority Register (DMA_DCHPRI19)	8	R/W	See section	22.3.16/ 697
114	Channel n Priority Register (DMA_DCHPRI20)	8	R/W	See section	22.3.16/ 697
115	Channel n Priority Register (DMA_DCHPRI21)	8	R/W	See section	22.3.16/ 697
116	Channel n Priority Register (DMA_DCHPRI22)	8	R/W	See section	22.3.16/ 697
117	Channel n Priority Register (DMA_DCHPRI23)	8	R/W	See section	22.3.16/ 697
118	Channel n Priority Register (DMA_DCHPRI24)	8	R/W	See section	22.3.16/ 697
119	Channel n Priority Register (DMA_DCHPRI25)	8	R/W	See section	22.3.16/ 697
11A	Channel n Priority Register (DMA_DCHPRI26)	8	R/W	See section	22.3.16/ 697
11B	Channel n Priority Register (DMA_DCHPRI27)	8	R/W	See section	22.3.16/ 697

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11C	Channel n Priority Register (DMA_DCHPRI28)	8	R/W	See section	22.3.16/ 697
11D	Channel n Priority Register (DMA_DCHPRI29)	8	R/W	See section	22.3.16/ 697
11E	Channel n Priority Register (DMA_DCHPRI30)	8	R/W	See section	22.3.16/ 697
11F	Channel n Priority Register (DMA_DCHPRI31)	8	R/W	See section	22.3.16/ 697
140	Channel n Master ID Register (DMA_DCHMID0)	8	R/W	00h	22.3.17/ 698
141	Channel n Master ID Register (DMA_DCHMID1)	8	R/W	00h	22.3.17/ 698
142	Channel n Master ID Register (DMA_DCHMID2)	8	R/W	00h	22.3.17/ 698
143	Channel n Master ID Register (DMA_DCHMID3)	8	R/W	00h	22.3.17/ 698
144	Channel n Master ID Register (DMA_DCHMID4)	8	R/W	00h	22.3.17/ 698
145	Channel n Master ID Register (DMA_DCHMID5)	8	R/W	00h	22.3.17/ 698
146	Channel n Master ID Register (DMA_DCHMID6)	8	R/W	00h	22.3.17/ 698
147	Channel n Master ID Register (DMA_DCHMID7)	8	R/W	00h	22.3.17/ 698
148	Channel n Master ID Register (DMA_DCHMID8)	8	R/W	00h	22.3.17/ 698
149	Channel n Master ID Register (DMA_DCHMID9)	8	R/W	00h	22.3.17/ 698
14A	Channel n Master ID Register (DMA_DCHMID10)	8	R/W	00h	22.3.17/ 698
14B	Channel n Master ID Register (DMA_DCHMID11)	8	R/W	00h	22.3.17/ 698
14C	Channel n Master ID Register (DMA_DCHMID12)	8	R/W	00h	22.3.17/ 698
14D	Channel n Master ID Register (DMA_DCHMID13)	8	R/W	00h	22.3.17/ 698
14E	Channel n Master ID Register (DMA_DCHMID14)	8	R/W	00h	22.3.17/ 698
14F	Channel n Master ID Register (DMA_DCHMID15)	8	R/W	00h	22.3.17/ 698
150	Channel n Master ID Register (DMA_DCHMID16)	8	R/W	00h	22.3.17/ 698
151	Channel n Master ID Register (DMA_DCHMID17)	8	R/W	00h	22.3.17/ 698

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
152	Channel n Master ID Register (DMA_DCHMID18)	8	R/W	00h	22.3.17/ 698
153	Channel n Master ID Register (DMA_DCHMID19)	8	R/W	00h	22.3.17/ 698
154	Channel n Master ID Register (DMA_DCHMID20)	8	R/W	00h	22.3.17/ 698
155	Channel n Master ID Register (DMA_DCHMID21)	8	R/W	00h	22.3.17/ 698
156	Channel n Master ID Register (DMA_DCHMID22)	8	R/W	00h	22.3.17/ 698
157	Channel n Master ID Register (DMA_DCHMID23)	8	R/W	00h	22.3.17/ 698
158	Channel n Master ID Register (DMA_DCHMID24)	8	R/W	00h	22.3.17/ 698
159	Channel n Master ID Register (DMA_DCHMID25)	8	R/W	00h	22.3.17/ 698
15A	Channel n Master ID Register (DMA_DCHMID26)	8	R/W	00h	22.3.17/ 698
15B	Channel n Master ID Register (DMA_DCHMID27)	8	R/W	00h	22.3.17/ 698
15C	Channel n Master ID Register (DMA_DCHMID28)	8	R/W	00h	22.3.17/ 698
15D	Channel n Master ID Register (DMA_DCHMID29)	8	R/W	00h	22.3.17/ 698
15E	Channel n Master ID Register (DMA_DCHMID30)	8	R/W	00h	22.3.17/ 698
15F	Channel n Master ID Register (DMA_DCHMID31)	8	R/W	00h	22.3.17/ 698
1000	TCD Source Address (DMA_TCD0_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1004	TCD Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	0000h	22.3.19/ 699
1006	TCD Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	0000h	22.3.20/ 700
1008	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1008	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1008	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
100C	TCD Last Source Address Adjustment (DMA_TCD0_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1010	TCD Destination Address (DMA_TCD0_DADDR)	32	R/W	0000_0000h	22.3.25/ 704

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1014	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1014	DMA_TCD0_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1016	TCD Signed Destination Address Offset (DMA_TCD0_DOFF)	16	R/W	0000h	22.3.28/ 707
1018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
101E	TCD Control and Status (DMA_TCD0_CSR)	16	R/W	0000h	22.3.32/ 710
1020	TCD Source Address (DMA_TCD1_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1024	TCD Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	0000h	22.3.19/ 699
1026	TCD Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	0000h	22.3.20/ 700
1028	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1028	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1028	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
102C	TCD Last Source Address Adjustment (DMA_TCD1_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1030	TCD Destination Address (DMA_TCD1_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1034	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1034	DMA_TCD1_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1036	TCD Signed Destination Address Offset (DMA_TCD1_DOFF)	16	R/W	0000h	22.3.28/ 707
1038	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
103C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
103C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
103E	TCD Control and Status (DMA_TCD1_CSR)	16	R/W	0000h	22.3.32/ 710
1040	TCD Source Address (DMA_TCD2_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1044	TCD Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	0000h	22.3.19/ 699
1046	TCD Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	0000h	22.3.20/ 700
1048	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1048	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1048	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
104C	TCD Last Source Address Adjustment (DMA_TCD2_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1050	TCD Destination Address (DMA_TCD2_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1054	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1054	DMA_TCD2_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1056	TCD Signed Destination Address Offset (DMA_TCD2_DOFF)	16	R/W	0000h	22.3.28/ 707
1058	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
105C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
105C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
105E	TCD Control and Status (DMA_TCD2_CSR)	16	R/W	0000h	22.3.32/ 710
1060	TCD Source Address (DMA_TCD3_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1064	TCD Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	0000h	22.3.19/ 699
1066	TCD Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	0000h	22.3.20/ 700
1068	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD3_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1068	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1068	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
106C	TCD Last Source Address Adjustment (DMA_TCD3_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1070	TCD Destination Address (DMA_TCD3_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1074	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1074	DMA_TCD3_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1076	TCD Signed Destination Address Offset (DMA_TCD3_DOFF)	16	R/W	0000h	22.3.28/ 707
1078	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
107C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
107C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
107E	TCD Control and Status (DMA_TCD3_CSR)	16	R/W	0000h	22.3.32/ 710
1080	TCD Source Address (DMA_TCD4_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1084	TCD Transfer Attributes (DMA_TCD4_ATTR)	16	R/W	0000h	22.3.19/ 699
1086	TCD Signed Source Address Offset (DMA_TCD4_SOFF)	16	R/W	0000h	22.3.20/ 700
1088	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD4_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1088	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD4_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1088	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD4_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
108C	TCD Last Source Address Adjustment (DMA_TCD4_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1090	TCD Destination Address (DMA_TCD4_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1094	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1094	DMA_TCD4_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1096	TCD Signed Destination Address Offset (DMA_TCD4_DOFF)	16	R/W	0000h	22.3.28/ 707
1098	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD4_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
109C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
109C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
109E	TCD Control and Status (DMA_TCD4_CSR)	16	R/W	0000h	22.3.32/ 710
10A0	TCD Source Address (DMA_TCD5_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
10A4	TCD Transfer Attributes (DMA_TCD5_ATTR)	16	R/W	0000h	22.3.19/ 699
10A6	TCD Signed Source Address Offset (DMA_TCD5_SOFF)	16	R/W	0000h	22.3.20/ 700
10A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD5_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
10A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD5_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
10A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD5_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
10AC	TCD Last Source Address Adjustment (DMA_TCD5_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
10B0	TCD Destination Address (DMA_TCD5_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
10B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
10B4	DMA_TCD5_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
10B6	TCD Signed Destination Address Offset (DMA_TCD5_DOFF)	16	R/W	0000h	22.3.28/ 707
10B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD5_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
10BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
10BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
10BE	TCD Control and Status (DMA_TCD5_CSR)	16	R/W	0000h	22.3.32/ 710
10C0	TCD Source Address (DMA_TCD6_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
10C4	TCD Transfer Attributes (DMA_TCD6_ATTR)	16	R/W	0000h	22.3.19/ 699
10C6	TCD Signed Source Address Offset (DMA_TCD6_SOFF)	16	R/W	0000h	22.3.20/ 700

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD6_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
10C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD6_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
10C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD6_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
10CC	TCD Last Source Address Adjustment (DMA_TCD6_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
10D0	TCD Destination Address (DMA_TCD6_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
10D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
10D4	DMA_TCD6_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
10D6	TCD Signed Destination Address Offset (DMA_TCD6_DOFF)	16	R/W	0000h	22.3.28/ 707
10D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD6_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
10DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
10DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
10DE	TCD Control and Status (DMA_TCD6_CSR)	16	R/W	0000h	22.3.32/ 710
10E0	TCD Source Address (DMA_TCD7_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
10E4	TCD Transfer Attributes (DMA_TCD7_ATTR)	16	R/W	0000h	22.3.19/ 699
10E6	TCD Signed Source Address Offset (DMA_TCD7_SOFF)	16	R/W	0000h	22.3.20/ 700
10E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD7_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
10E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD7_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
10E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD7_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
10EC	TCD Last Source Address Adjustment (DMA_TCD7_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
10F0	TCD Destination Address (DMA_TCD7_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
10F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10F4	DMA_TCD7_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
10F6	TCD Signed Destination Address Offset (DMA_TCD7_DOFF)	16	R/W	0000h	22.3.28/ 707
10F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD7_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
10FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
10FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
10FE	TCD Control and Status (DMA_TCD7_CSR)	16	R/W	0000h	22.3.32/ 710
1100	TCD Source Address (DMA_TCD8_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1104	TCD Transfer Attributes (DMA_TCD8_ATTR)	16	R/W	0000h	22.3.19/ 699
1106	TCD Signed Source Address Offset (DMA_TCD8_SOFF)	16	R/W	0000h	22.3.20/ 700
1108	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD8_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1108	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD8_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1108	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD8_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
110C	TCD Last Source Address Adjustment (DMA_TCD8_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1110	TCD Destination Address (DMA_TCD8_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1114	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1114	DMA_TCD8_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1116	TCD Signed Destination Address Offset (DMA_TCD8_DOFF)	16	R/W	0000h	22.3.28/ 707
1118	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD8_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
111C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
111C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD8_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
111E	TCD Control and Status (DMA_TCD8_CSR)	16	R/W	0000h	22.3.32/ 710

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1120	TCD Source Address (DMA_TCD9_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1124	TCD Transfer Attributes (DMA_TCD9_ATTR)	16	R/W	0000h	22.3.19/ 699
1126	TCD Signed Source Address Offset (DMA_TCD9_SOFF)	16	R/W	0000h	22.3.20/ 700
1128	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD9_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1128	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD9_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1128	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD9_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
112C	TCD Last Source Address Adjustment (DMA_TCD9_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1130	TCD Destination Address (DMA_TCD9_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1134	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1134	DMA_TCD9_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1136	TCD Signed Destination Address Offset (DMA_TCD9_DOFF)	16	R/W	0000h	22.3.28/ 707
1138	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD9_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
113C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
113C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD9_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
113E	TCD Control and Status (DMA_TCD9_CSR)	16	R/W	0000h	22.3.32/ 710
1140	TCD Source Address (DMA_TCD10_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1144	TCD Transfer Attributes (DMA_TCD10_ATTR)	16	R/W	0000h	22.3.19/ 699
1146	TCD Signed Source Address Offset (DMA_TCD10_SOFF)	16	R/W	0000h	22.3.20/ 700
1148	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD10_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1148	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD10_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1148	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD10_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
114C	TCD Last Source Address Adjustment (DMA_TCD10_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1150	TCD Destination Address (DMA_TCD10_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1154	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1154	DMA_TCD10_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1156	TCD Signed Destination Address Offset (DMA_TCD10_DOFF)	16	R/W	0000h	22.3.28/ 707
1158	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD10_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
115C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
115C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD10_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
115E	TCD Control and Status (DMA_TCD10_CSR)	16	R/W	0000h	22.3.32/ 710
1160	TCD Source Address (DMA_TCD11_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1164	TCD Transfer Attributes (DMA_TCD11_ATTR)	16	R/W	0000h	22.3.19/ 699
1166	TCD Signed Source Address Offset (DMA_TCD11_SOFF)	16	R/W	0000h	22.3.20/ 700
1168	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD11_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1168	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD11_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1168	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD11_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
116C	TCD Last Source Address Adjustment (DMA_TCD11_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1170	TCD Destination Address (DMA_TCD11_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1174	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1174	DMA_TCD11_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1176	TCD Signed Destination Address Offset (DMA_TCD11_DOFF)	16	R/W	0000h	22.3.28/ 707
1178	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD11_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
117C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
117C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD11_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
117E	TCD Control and Status (DMA_TCD11_CSR)	16	R/W	0000h	22.3.32/ 710
1180	TCD Source Address (DMA_TCD12_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1184	TCD Transfer Attributes (DMA_TCD12_ATTR)	16	R/W	0000h	22.3.19/ 699
1186	TCD Signed Source Address Offset (DMA_TCD12_SOFF)	16	R/W	0000h	22.3.20/ 700
1188	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD12_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1188	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD12_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1188	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD12_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
118C	TCD Last Source Address Adjustment (DMA_TCD12_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1190	TCD Destination Address (DMA_TCD12_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1194	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1194	DMA_TCD12_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1196	TCD Signed Destination Address Offset (DMA_TCD12_DOFF)	16	R/W	0000h	22.3.28/ 707
1198	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD12_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
119C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
119C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD12_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
119E	TCD Control and Status (DMA_TCD12_CSR)	16	R/W	0000h	22.3.32/ 710
11A0	TCD Source Address (DMA_TCD13_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
11A4	TCD Transfer Attributes (DMA_TCD13_ATTR)	16	R/W	0000h	22.3.19/ 699

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11A6	TCD Signed Source Address Offset (DMA_TCD13_SOFF)	16	R/W	0000h	22.3.20/ 700
11A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD13_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
11A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD13_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
11A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD13_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
11AC	TCD Last Source Address Adjustment (DMA_TCD13_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
11B0	TCD Destination Address (DMA_TCD13_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
11B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
11B4	DMA_TCD13_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
11B6	TCD Signed Destination Address Offset (DMA_TCD13_DOFF)	16	R/W	0000h	22.3.28/ 707
11B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD13_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
11BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
11BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD13_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
11BE	TCD Control and Status (DMA_TCD13_CSR)	16	R/W	0000h	22.3.32/ 710
11C0	TCD Source Address (DMA_TCD14_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
11C4	TCD Transfer Attributes (DMA_TCD14_ATTR)	16	R/W	0000h	22.3.19/ 699
11C6	TCD Signed Source Address Offset (DMA_TCD14_SOFF)	16	R/W	0000h	22.3.20/ 700
11C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD14_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
11C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD14_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
11C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD14_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
11CC	TCD Last Source Address Adjustment (DMA_TCD14_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
11D0	TCD Destination Address (DMA_TCD14_DADDR)	32	R/W	0000_0000h	22.3.25/ 704

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
11D4	DMA_TCD14_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
11D6	TCD Signed Destination Address Offset (DMA_TCD14_DOFF)	16	R/W	0000h	22.3.28/ 707
11D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD14_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
11DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
11DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD14_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
11DE	TCD Control and Status (DMA_TCD14_CSR)	16	R/W	0000h	22.3.32/ 710
11E0	TCD Source Address (DMA_TCD15_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
11E4	TCD Transfer Attributes (DMA_TCD15_ATTR)	16	R/W	0000h	22.3.19/ 699
11E6	TCD Signed Source Address Offset (DMA_TCD15_SOFF)	16	R/W	0000h	22.3.20/ 700
11E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD15_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
11E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD15_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
11E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD15_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
11EC	TCD Last Source Address Adjustment (DMA_TCD15_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
11F0	TCD Destination Address (DMA_TCD15_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
11F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
11F4	DMA_TCD15_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
11F6	TCD Signed Destination Address Offset (DMA_TCD15_DOFF)	16	R/W	0000h	22.3.28/ 707
11F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD15_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
11FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD15_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
11FE	TCD Control and Status (DMA_TCD15_CSR)	16	R/W	0000h	22.3.32/ 710
1200	TCD Source Address (DMA_TCD16_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1204	TCD Transfer Attributes (DMA_TCD16_ATTR)	16	R/W	0000h	22.3.19/ 699
1206	TCD Signed Source Address Offset (DMA_TCD16_SOFF)	16	R/W	0000h	22.3.20/ 700
1208	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD16_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1208	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD16_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1208	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD16_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
120C	TCD Last Source Address Adjustment (DMA_TCD16_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1210	TCD Destination Address (DMA_TCD16_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1214	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1214	DMA_TCD16_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1216	TCD Signed Destination Address Offset (DMA_TCD16_DOFF)	16	R/W	0000h	22.3.28/ 707
1218	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD16_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
121C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
121C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD16_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
121E	TCD Control and Status (DMA_TCD16_CSR)	16	R/W	0000h	22.3.32/ 710
1220	TCD Source Address (DMA_TCD17_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1224	TCD Transfer Attributes (DMA_TCD17_ATTR)	16	R/W	0000h	22.3.19/ 699
1226	TCD Signed Source Address Offset (DMA_TCD17_SOFF)	16	R/W	0000h	22.3.20/ 700
1228	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD17_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1228	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD17_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1228	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD17_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
122C	TCD Last Source Address Adjustment (DMA_TCD17_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1230	TCD Destination Address (DMA_TCD17_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1234	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1234	DMA_TCD17_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1236	TCD Signed Destination Address Offset (DMA_TCD17_DOFF)	16	R/W	0000h	22.3.28/ 707
1238	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD17_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
123C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
123C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD17_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
123E	TCD Control and Status (DMA_TCD17_CSR)	16	R/W	0000h	22.3.32/ 710
1240	TCD Source Address (DMA_TCD18_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1244	TCD Transfer Attributes (DMA_TCD18_ATTR)	16	R/W	0000h	22.3.19/ 699
1246	TCD Signed Source Address Offset (DMA_TCD18_SOFF)	16	R/W	0000h	22.3.20/ 700
1248	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD18_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1248	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD18_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1248	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD18_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
124C	TCD Last Source Address Adjustment (DMA_TCD18_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1250	TCD Destination Address (DMA_TCD18_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1254	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1254	DMA_TCD18_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1256	TCD Signed Destination Address Offset (DMA_TCD18_DOFF)	16	R/W	0000h	22.3.28/ 707
1258	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD18_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
125C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
125C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD18_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
125E	TCD Control and Status (DMA_TCD18_CSR)	16	R/W	0000h	22.3.32/ 710
1260	TCD Source Address (DMA_TCD19_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1264	TCD Transfer Attributes (DMA_TCD19_ATTR)	16	R/W	0000h	22.3.19/ 699
1266	TCD Signed Source Address Offset (DMA_TCD19_SOFF)	16	R/W	0000h	22.3.20/ 700
1268	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD19_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1268	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD19_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1268	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD19_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
126C	TCD Last Source Address Adjustment (DMA_TCD19_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1270	TCD Destination Address (DMA_TCD19_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1274	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1274	DMA_TCD19_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1276	TCD Signed Destination Address Offset (DMA_TCD19_DOFF)	16	R/W	0000h	22.3.28/ 707
1278	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD19_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
127C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
127C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD19_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
127E	TCD Control and Status (DMA_TCD19_CSR)	16	R/W	0000h	22.3.32/ 710

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1280	TCD Source Address (DMA_TCD20_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1284	TCD Transfer Attributes (DMA_TCD20_ATTR)	16	R/W	0000h	22.3.19/ 699
1286	TCD Signed Source Address Offset (DMA_TCD20_SOFF)	16	R/W	0000h	22.3.20/ 700
1288	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD20_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1288	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD20_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1288	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD20_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
128C	TCD Last Source Address Adjustment (DMA_TCD20_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1290	TCD Destination Address (DMA_TCD20_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1294	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1294	DMA_TCD20_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1296	TCD Signed Destination Address Offset (DMA_TCD20_DOFF)	16	R/W	0000h	22.3.28/ 707
1298	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD20_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
129C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
129C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD20_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
129E	TCD Control and Status (DMA_TCD20_CSR)	16	R/W	0000h	22.3.32/ 710
12A0	TCD Source Address (DMA_TCD21_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
12A4	TCD Transfer Attributes (DMA_TCD21_ATTR)	16	R/W	0000h	22.3.19/ 699
12A6	TCD Signed Source Address Offset (DMA_TCD21_SOFF)	16	R/W	0000h	22.3.20/ 700
12A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD21_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
12A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD21_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
12A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD21_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12AC	TCD Last Source Address Adjustment (DMA_TCD21_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
12B0	TCD Destination Address (DMA_TCD21_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
12B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
12B4	DMA_TCD21_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
12B6	TCD Signed Destination Address Offset (DMA_TCD21_DOFF)	16	R/W	0000h	22.3.28/ 707
12B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD21_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
12BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
12BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD21_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
12BE	TCD Control and Status (DMA_TCD21_CSR)	16	R/W	0000h	22.3.32/ 710
12C0	TCD Source Address (DMA_TCD22_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
12C4	TCD Transfer Attributes (DMA_TCD22_ATTR)	16	R/W	0000h	22.3.19/ 699
12C6	TCD Signed Source Address Offset (DMA_TCD22_SOFF)	16	R/W	0000h	22.3.20/ 700
12C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD22_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
12C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD22_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
12C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD22_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
12CC	TCD Last Source Address Adjustment (DMA_TCD22_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
12D0	TCD Destination Address (DMA_TCD22_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
12D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
12D4	DMA_TCD22_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
12D6	TCD Signed Destination Address Offset (DMA_TCD22_DOFF)	16	R/W	0000h	22.3.28/ 707
12D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD22_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
12DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD22_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
12DE	TCD Control and Status (DMA_TCD22_CSR)	16	R/W	0000h	22.3.32/ 710
12E0	TCD Source Address (DMA_TCD23_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
12E4	TCD Transfer Attributes (DMA_TCD23_ATTR)	16	R/W	0000h	22.3.19/ 699
12E6	TCD Signed Source Address Offset (DMA_TCD23_SOFF)	16	R/W	0000h	22.3.20/ 700
12E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD23_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
12E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD23_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
12E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD23_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
12EC	TCD Last Source Address Adjustment (DMA_TCD23_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
12F0	TCD Destination Address (DMA_TCD23_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
12F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
12F4	DMA_TCD23_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
12F6	TCD Signed Destination Address Offset (DMA_TCD23_DOFF)	16	R/W	0000h	22.3.28/ 707
12F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD23_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
12FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
12FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD23_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
12FE	TCD Control and Status (DMA_TCD23_CSR)	16	R/W	0000h	22.3.32/ 710
1300	TCD Source Address (DMA_TCD24_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1304	TCD Transfer Attributes (DMA_TCD24_ATTR)	16	R/W	0000h	22.3.19/ 699

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1306	TCD Signed Source Address Offset (DMA_TCD24_SOFF)	16	R/W	0000h	22.3.20/ 700
1308	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD24_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1308	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD24_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1308	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD24_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
130C	TCD Last Source Address Adjustment (DMA_TCD24_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1310	TCD Destination Address (DMA_TCD24_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1314	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1314	DMA_TCD24_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1316	TCD Signed Destination Address Offset (DMA_TCD24_DOFF)	16	R/W	0000h	22.3.28/ 707
1318	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD24_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
131C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
131C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD24_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
131E	TCD Control and Status (DMA_TCD24_CSR)	16	R/W	0000h	22.3.32/ 710
1320	TCD Source Address (DMA_TCD25_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1324	TCD Transfer Attributes (DMA_TCD25_ATTR)	16	R/W	0000h	22.3.19/ 699
1326	TCD Signed Source Address Offset (DMA_TCD25_SOFF)	16	R/W	0000h	22.3.20/ 700
1328	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD25_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1328	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD25_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1328	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD25_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
132C	TCD Last Source Address Adjustment (DMA_TCD25_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1330	TCD Destination Address (DMA_TCD25_DADDR)	32	R/W	0000_0000h	22.3.25/ 704

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1334	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1334	DMA_TCD25_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1336	TCD Signed Destination Address Offset (DMA_TCD25_DOFF)	16	R/W	0000h	22.3.28/ 707
1338	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD25_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
133C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
133C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD25_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
133E	TCD Control and Status (DMA_TCD25_CSR)	16	R/W	0000h	22.3.32/ 710
1340	TCD Source Address (DMA_TCD26_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1344	TCD Transfer Attributes (DMA_TCD26_ATTR)	16	R/W	0000h	22.3.19/ 699
1346	TCD Signed Source Address Offset (DMA_TCD26_SOFF)	16	R/W	0000h	22.3.20/ 700
1348	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD26_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1348	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD26_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1348	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD26_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
134C	TCD Last Source Address Adjustment (DMA_TCD26_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1350	TCD Destination Address (DMA_TCD26_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1354	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1354	DMA_TCD26_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1356	TCD Signed Destination Address Offset (DMA_TCD26_DOFF)	16	R/W	0000h	22.3.28/ 707
1358	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD26_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
135C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
135C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD26_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
135E	TCD Control and Status (DMA_TCD26_CSR)	16	R/W	0000h	22.3.32/ 710
1360	TCD Source Address (DMA_TCD27_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1364	TCD Transfer Attributes (DMA_TCD27_ATTR)	16	R/W	0000h	22.3.19/ 699
1366	TCD Signed Source Address Offset (DMA_TCD27_SOFF)	16	R/W	0000h	22.3.20/ 700
1368	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD27_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
1368	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD27_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1368	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD27_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
136C	TCD Last Source Address Adjustment (DMA_TCD27_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1370	TCD Destination Address (DMA_TCD27_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1374	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1374	DMA_TCD27_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1376	TCD Signed Destination Address Offset (DMA_TCD27_DOFF)	16	R/W	0000h	22.3.28/ 707
1378	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD27_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
137C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
137C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD27_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
137E	TCD Control and Status (DMA_TCD27_CSR)	16	R/W	0000h	22.3.32/ 710
1380	TCD Source Address (DMA_TCD28_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
1384	TCD Transfer Attributes (DMA_TCD28_ATTR)	16	R/W	0000h	22.3.19/ 699
1386	TCD Signed Source Address Offset (DMA_TCD28_SOFF)	16	R/W	0000h	22.3.20/ 700
1388	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD28_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1388	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD28_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
1388	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD28_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
138C	TCD Last Source Address Adjustment (DMA_TCD28_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
1390	TCD Destination Address (DMA_TCD28_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
1394	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
1394	DMA_TCD28_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
1396	TCD Signed Destination Address Offset (DMA_TCD28_DOFF)	16	R/W	0000h	22.3.28/ 707
1398	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD28_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
139C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
139C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD28_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
139E	TCD Control and Status (DMA_TCD28_CSR)	16	R/W	0000h	22.3.32/ 710
13A0	TCD Source Address (DMA_TCD29_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
13A4	TCD Transfer Attributes (DMA_TCD29_ATTR)	16	R/W	0000h	22.3.19/ 699
13A6	TCD Signed Source Address Offset (DMA_TCD29_SOFF)	16	R/W	0000h	22.3.20/ 700
13A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD29_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
13A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD29_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
13A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD29_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
13AC	TCD Last Source Address Adjustment (DMA_TCD29_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
13B0	TCD Destination Address (DMA_TCD29_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
13B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
13B4	DMA_TCD29_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13B6	TCD Signed Destination Address Offset (DMA_TCD29_DOFF)	16	R/W	0000h	22.3.28/ 707
13B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD29_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
13BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
13BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD29_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
13BE	TCD Control and Status (DMA_TCD29_CSR)	16	R/W	0000h	22.3.32/ 710
13C0	TCD Source Address (DMA_TCD30_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
13C4	TCD Transfer Attributes (DMA_TCD30_ATTR)	16	R/W	0000h	22.3.19/ 699
13C6	TCD Signed Source Address Offset (DMA_TCD30_SOFF)	16	R/W	0000h	22.3.20/ 700
13C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD30_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
13C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD30_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
13C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD30_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
13CC	TCD Last Source Address Adjustment (DMA_TCD30_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
13D0	TCD Destination Address (DMA_TCD30_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
13D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
13D4	DMA_TCD30_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
13D6	TCD Signed Destination Address Offset (DMA_TCD30_DOFF)	16	R/W	0000h	22.3.28/ 707
13D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD30_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
13DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
13DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD30_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
13DE	TCD Control and Status (DMA_TCD30_CSR)	16	R/W	0000h	22.3.32/ 710

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13E0	TCD Source Address (DMA_TCD31_SADDR)	32	R/W	0000_0000h	22.3.18/ 699
13E4	TCD Transfer Attributes (DMA_TCD31_ATTR)	16	R/W	0000h	22.3.19/ 699
13E6	TCD Signed Source Address Offset (DMA_TCD31_SOFF)	16	R/W	0000h	22.3.20/ 700
13E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD31_NBYTES_MLNO)	32	R/W	0000_0000h	22.3.21/ 701
13E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD31_NBYTES_MLOFFNO)	32	R/W	0000_0000h	22.3.22/ 701
13E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD31_NBYTES_MLOFFYES)	32	R/W	0000_0000h	22.3.23/ 703
13EC	TCD Last Source Address Adjustment (DMA_TCD31_SLAST)	32	R/W	0000_0000h	22.3.24/ 704
13F0	TCD Destination Address (DMA_TCD31_DADDR)	32	R/W	0000_0000h	22.3.25/ 704
13F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_CITER_ELINKYES)	16	R/W	0000h	22.3.26/ 705
13F4	DMA_TCD31_CITER_ELINKNO	16	R/W	0000h	22.3.27/ 706
13F6	TCD Signed Destination Address Offset (DMA_TCD31_DOFF)	16	R/W	0000h	22.3.28/ 707
13F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD31_DLASTSGA)	32	R/W	0000_0000h	22.3.29/ 707
13FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_BITER_ELINKYES)	16	R/W	0000h	22.3.30/ 708
13FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD31_BITER_ELINKNO)	16	R/W	0000h	22.3.31/ 709
13FE	TCD Control and Status (DMA_TCD31_CSR)	16	R/W	0000h	22.3.32/ 710

22.3.1 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA. The DMA arbitrates channel service requests in two groups of 16 channels each:

- Group 1 contains channels 31-16
- Group 0 contains channels 15-0

Arbitration within a group can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels within each group are cycled through (from high to low channel number) without regard to priority.

NOTE

> For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first, where priority level 1 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to any channel service requests occurring; otherwise, a configuration error will be reported. For group round robin arbitration, the group priorities are ignored and the groups are cycled through (from high to low group number) without regard to priority.

Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														CX	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					GRP1PRI	0	GRP0PRI	EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	Reserved
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

DMA_CR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CX	Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
24 EMLM	Enable Minor Loop Mapping 0 Disabled. TCDn.word2 is defined as a 32-bit NBYTES field. 1 Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
25 CLM	Continuous Link Mode **NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, e.g., if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA Operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt On Error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable Round Robin Group Arbitration 0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
29 ERCA	Enable Round Robin Channel Arbitration 0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.
30 EDBG	Enable Debug 0 When in debug mode, the DMA continues to operate. 1 When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared.
31 Reserved	This field is reserved. Reserved

22.3.2 Error Status Register (DMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor, or
  - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle

- A bit in the ES shows an uncorrectable error occurred while the device has ECC protection on the TCD SRAM
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See the Error Reporting and Handling section for more details.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0													UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	0	ERRCHN					SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ES field descriptions

Field	Description
0 VLD	Logical OR of all ERR status bits 0 No ERR bits are set. 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
1–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 UCE	Uncorrectable ECC error during channel execution. The UCE bit is set when an uncorrectable ECC error occurs on an access generated by the DMA only. If a CPU access to the TCD causes an uncorrectable ECC error, that access will receive a bus error response. **NOTE:** When the eDMA sees a RAM error on an IPS access (when the user is accessing a TCD), the error is reported as a bus abort. When the dma_engine (the execution engine is accessing a TCD) receives a RAM error, it is recorded in the Error Status register, DMA_ES[UCE], along with the channel number. 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error
15 ECX	Transfer Canceled 0 No canceled transfers 1 The last recorded entry was a canceled transfer by the error cancel transfer input
16 GPE	Group Priority Error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
17 CPE	Channel Priority Error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities within a group. Channel priorities within a group are not unique.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error, excluding GPE and CPE errors, or last recorded error canceled transfer.
24 SAE	Source Address Error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
25 SOE	Source Offset Error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
26 DAE	Destination Address Error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
27 DOE	Destination Offset Error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
28 NCE	NBYTES/CITER Configuration Error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or • TCDn_CITER[CITER] is equal to zero, or • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]
29 SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary.
30 SBE	Source Bus Error 0 No source bus error 1 The last recorded error was a bus error on a source read

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
31 DBE	Destination Bus Error 0 No destination bus error 1 The last recorded error was a bus error on a destination write

22.3.3 Enable Request Register (DMA_ERQ)

The ERQ register provides a bit map for the 32 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERQ field descriptions

Field	Description
0 ERQ31	Enable DMA Request 31 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ28	Enable DMA Request 28 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ25	Enable DMA Request 25 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ23	Enable DMA Request 23 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ22	Enable DMA Request 22 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ21	Enable DMA Request 21 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ16	Enable DMA Request 16 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ15	Enable DMA Request 15 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ11	Enable DMA Request 11 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ10	Enable DMA Request 10 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ9	Enable DMA Request 9 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ8	Enable DMA Request 8 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

22.3.4 Enable Error Interrupt Register (DMA_EEI)

The EEI register provides a bit map for the 32 channels to enable the error interrupt signal for each channel. The state of any given channel’s error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEI field descriptions

Field	Description
0 EEI31	Enable Error Interrupt 31 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI30	Enable Error Interrupt 30 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable Error Interrupt 29 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable Error Interrupt 28 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI27	Enable Error Interrupt 27 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable Error Interrupt 26 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable Error Interrupt 25 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable Error Interrupt 24 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEI field descriptions (continued)

Field	Description
8 EEI23	Enable Error Interrupt 23 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI22	Enable Error Interrupt 22 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable Error Interrupt 21 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable Error Interrupt 20 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI19	Enable Error Interrupt 19 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable Error Interrupt 18 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable Error Interrupt 17 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable Error Interrupt 16 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI15	Enable Error Interrupt 15 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable Error Interrupt 14 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI13	Enable Error Interrupt 13 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI12	Enable Error Interrupt 12 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEI field descriptions (continued)

Field	Description
20 EEI11	Enable Error Interrupt 11 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable Error Interrupt 10 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable Error Interrupt 9 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI8	Enable Error Interrupt 8 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI7	Enable Error Interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable Error Interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable Error Interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable Error Interrupt 4 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI3	Enable Error Interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable Error Interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI1	Enable Error Interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable Error Interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

22.3.5 Set Enable Request Register (DMA_SERQ)

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	SAER	0	SERQ				
Reset	0	0	0	0	0	0	0	0

DMA_SERQ field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAER	Set All Enable Requests 0 Set only the ERQ bit specified in the SERQ field 1 Set all bits in ERQ
2 Reserved	This field is reserved.
3–7 SERQ	Set Enable Request Sets the corresponding bit in ERQ.

22.3.6 Clear Enable Request Register (DMA_CERQ)

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ to be cleared, disabling all DMA request inputs. If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 19h offset = 19h

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CAER	0	CERQ				
Reset	0	0	0	0	0	0	0	0

DMA_CERQ field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAER	Clear All Enable Requests 0 Clear only the ERQ bit specified in the CERQ field 1 Clear all bits in ERQ
2 Reserved	This field is reserved.
3–7 CERQ	Clear Enable Request Clears the corresponding bit in ERQ.

22.3.7 Set Enable Error Interrupt Register (DMA_SEEI)

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	SAEE	0	SEEI				
Reset	0	0	0	0	0	0	0	0

DMA_SEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAEE	Sets All Enable Error Interrupts 0 Set only the EEI bit specified in the SEEI field. 1 Sets all bits in EEI
2 Reserved	This field is reserved.
3–7 SEEI	Set Enable Error Interrupt Sets the corresponding bit in EEI

22.3.8 Clear Enable Error Interrupt Register (DMA_CEEI)

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI contents to be cleared, disabling all DMA request inputs. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Bh offset = 1Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CAEE	0	CEEI				
Reset	0	0	0	0	0	0	0	0

DMA_CEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAEE	Clear All Enable Error Interrupts 0 Clear only the EEI bit specified in the CEEI field 1 Clear all bits in EEI
2 Reserved	This field is reserved.
3–7 CEEI	Clear Enable Error Interrupt Clears the corresponding bit in EEI

22.3.9 Clear Interrupt Request Register (DMA_CINT)

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CAIR	0	CINT				
Reset	0	0	0	0	0	0	0	0

DMA_CINT field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAIR	Clear All Interrupt Requests 0 Clear only the INT bit specified in the CINT field 1 Clear all bits in INT
2 Reserved	This field is reserved.
3–7 CINT	Clear Interrupt Request Clears the corresponding bit in INT

22.3.10 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Dh offset = 1Dh

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CAEI	0	CERR				
Reset	0	0	0	0	0	0	0	0

DMA_CERR field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAEI	Clear All Error Indicators 0 Clear only the ERR bit specified in the CERR field 1 Clear all bits in ERR
2 Reserved	This field is reserved.
3–7 CERR	Clear Error Indicator Clears the corresponding bit in ERR

22.3.11 Set START Bit Register (DMA_SSRT)

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	SAST	0	SSRT				
Reset	0	0	0	0	0	0	0	0

DMA_SSRT field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAST	Set All START Bits (activates all channels) 0 Set only the TCDn_CSR[START] bit specified in the SSRT field 1 Set all bits in TCDn_CSR[START]
2 Reserved	This field is reserved.
3–7 SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

22.3.12 Clear DONE Status Bit Register (DMA_CDNE)

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Fh offset = 1Fh

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CADN	0	CDNE				
Reset	0	0	0	0	0	0	0	0

DMA_CDNE field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CADN	Clears All DONE Bits 0 Clears only the TCDn_CSR[DONE] bit specified in the CDNE field 1 Clears all bits in TCDn_CSR[DONE]
2 Reserved	This field is reserved.
3–7 CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

22.3.13 Interrupt Request Register (DMA_INT)

The INT register provides a bit map for the 32 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INT field descriptions

Field	Description
0 INT31	Interrupt Request 31

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT30	Interrupt Request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT29	Interrupt Request 29 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT28	Interrupt Request 28 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT27	Interrupt Request 27 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT26	Interrupt Request 26 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT25	Interrupt Request 25 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT24	Interrupt Request 24 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT23	Interrupt Request 23 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT22	Interrupt Request 22 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT21	Interrupt Request 21 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT20	Interrupt Request 20 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT19	Interrupt Request 19

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT18	Interrupt Request 18 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT17	Interrupt Request 17 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT16	Interrupt Request 16 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT15	Interrupt Request 15 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT14	Interrupt Request 14 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT13	Interrupt Request 13 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT12	Interrupt Request 12 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT11	Interrupt Request 11 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT10	Interrupt Request 10 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT9	Interrupt Request 9 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT8	Interrupt Request 8 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT7	Interrupt Request 7

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT6	Interrupt Request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT5	Interrupt Request 5 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT4	Interrupt Request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT3	Interrupt Request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT2	Interrupt Request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT1	Interrupt Request 1 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT0	Interrupt Request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

22.3.14 Error Register (DMA_ERR)

The ERR provides a bit map for the 32 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16 and 32 channels to form several group error interrupt requests, which are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software’s responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERR field descriptions

Field	Description
0 ERR31	Error In Channel 31 0 An error in this channel has not occurred 1 An error in this channel has occurred
1 ERR30	Error In Channel 30 0 An error in this channel has not occurred 1 An error in this channel has occurred
2 ERR29	Error In Channel 29 0 An error in this channel has not occurred 1 An error in this channel has occurred
3 ERR28	Error In Channel 28 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
4 ERR27	Error In Channel 27 0 An error in this channel has not occurred 1 An error in this channel has occurred
5 ERR26	Error In Channel 26 0 An error in this channel has not occurred 1 An error in this channel has occurred
6 ERR25	Error In Channel 25 0 An error in this channel has not occurred 1 An error in this channel has occurred
7 ERR24	Error In Channel 24 0 An error in this channel has not occurred 1 An error in this channel has occurred
8 ERR23	Error In Channel 23 0 An error in this channel has not occurred 1 An error in this channel has occurred
9 ERR22	Error In Channel 22 0 An error in this channel has not occurred 1 An error in this channel has occurred
10 ERR21	Error In Channel 21 0 An error in this channel has not occurred 1 An error in this channel has occurred
11 ERR20	Error In Channel 20 0 An error in this channel has not occurred 1 An error in this channel has occurred
12 ERR19	Error In Channel 19 0 An error in this channel has not occurred 1 An error in this channel has occurred
13 ERR18	Error In Channel 18 0 An error in this channel has not occurred 1 An error in this channel has occurred
14 ERR17	Error In Channel 17 0 An error in this channel has not occurred 1 An error in this channel has occurred
15 ERR16	Error In Channel 16 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
16 ERR15	Error In Channel 15 0 An error in this channel has not occurred 1 An error in this channel has occurred
17 ERR14	Error In Channel 14 0 An error in this channel has not occurred 1 An error in this channel has occurred
18 ERR13	Error In Channel 13 0 An error in this channel has not occurred 1 An error in this channel has occurred
19 ERR12	Error In Channel 12 0 An error in this channel has not occurred 1 An error in this channel has occurred
20 ERR11	Error In Channel 11 0 An error in this channel has not occurred 1 An error in this channel has occurred
21 ERR10	Error In Channel 10 0 An error in this channel has not occurred 1 An error in this channel has occurred
22 ERR9	Error In Channel 9 0 An error in this channel has not occurred 1 An error in this channel has occurred
23 ERR8	Error In Channel 8 0 An error in this channel has not occurred 1 An error in this channel has occurred
24 ERR7	Error In Channel 7 0 An error in this channel has not occurred 1 An error in this channel has occurred
25 ERR6	Error In Channel 6 0 An error in this channel has not occurred 1 An error in this channel has occurred
26 ERR5	Error In Channel 5 0 An error in this channel has not occurred 1 An error in this channel has occurred
27 ERR4	Error In Channel 4 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
28 ERR3	Error In Channel 3 0 An error in this channel has not occurred 1 An error in this channel has occurred
29 ERR2	Error In Channel 2 0 An error in this channel has not occurred 1 An error in this channel has occurred
30 ERR1	Error In Channel 1 0 An error in this channel has not occurred 1 An error in this channel has occurred
31 ERR0	Error In Channel 0 0 An error in this channel has not occurred 1 An error in this channel has occurred

22.3.15 Hardware Request Status Register (DMA_HRS)

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS9	HRS8	HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRS field descriptions

Field	Description
0 HRS31	Hardware Request Status Channel 31 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 31 is not present 1 A hardware service request for channel 31 is present
1 HRS30	Hardware Request Status Channel 30 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 30 is not present 1 A hardware service request for for channel 30 is present
2 HRS29	Hardware Request Status Channel 29 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 29 is not preset 1 A hardware service request for channel 29 is present
3 HRS28	Hardware Request Status Channel 28 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 28 is not present 1 A hardware service request for channel 28 is present
4 HRS27	Hardware Request Status Channel 27 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 27 is not present 1 A hardware service request for channel 27 is present
5 HRS26	Hardware Request Status Channel 26 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 26 is not present 1 A hardware service request for channel 26 is present
6 HRS25	Hardware Request Status Channel 25 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 25 is not present 1 A hardware service request for channel 25 is present
7 HRS24	Hardware Request Status Channel 24 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 24 is not present 1 A hardware service request for channel 24 is present
8 HRS23	Hardware Request Status Channel 23 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 23 is not present 1 A hardware service request for channel 23 is present
9 HRS22	Hardware Request Status Channel 22 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 22 is not present 1 A hardware service request for channel 22 is present
10 HRS21	Hardware Request Status Channel 21 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 21 is not present 1 A hardware service request for channel 21 is present
11 HRS20	Hardware Request Status Channel 20

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 20 is not present 1 A hardware service request for channel 20 is present
12 HRS19	Hardware Request Status Channel 19 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 19 is not present 1 A hardware service request for channel 19 is present
13 HRS18	Hardware Request Status Channel 18 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 18 is not present 1 A hardware service request for channel 18 is present
14 HRS17	Hardware Request Status Channel 17 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 17 is not present 1 A hardware service request for channel 17 is present
15 HRS16	Hardware Request Status Channel 16 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 16 is not present 1 A hardware service request for channel 16 is present
16 HRS15	Hardware Request Status Channel 15 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 15 is not present 1 A hardware service request for channel 15 is present
17 HRS14	Hardware Request Status Channel 14 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 14 is not present 1 A hardware service request for channel 14 is present
18 HRS13	Hardware Request Status Channel 13 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 13 is not present 1 A hardware service request for channel 13 is present
19 HRS12	Hardware Request Status Channel 12 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 12 is not present 1 A hardware service request for channel 12 is present
20 HRS11	Hardware Request Status Channel 11 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 11 is not present 1 A hardware service request for channel 11 is present
21 HRS10	Hardware Request Status Channel 10 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 10 is not present 1 A hardware service request for channel 10 is present
22 HRS9	Hardware Request Status Channel 9 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 9 is not present 1 A hardware service request for channel 9 is present
23 HRS8	Hardware Request Status Channel 8 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 8 is not present 1 A hardware service request for channel 8 is present
24 HRS7	Hardware Request Status Channel 7

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 7 is not present 1 A hardware service request for channel 7 is present
25 HRS6	Hardware Request Status Channel 6 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 6 is not present 1 A hardware service request for channel 6 is present
26 HRS5	Hardware Request Status Channel 5 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 5 is not present 1 A hardware service request for channel 5 is present
27 HRS4	Hardware Request Status Channel 4 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 4 is not present 1 A hardware service request for channel 4 is present
28 HRS3	Hardware Request Status Channel 3 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 3 is not present 1 A hardware service request for channel 3 is present
29 HRS2	Hardware Request Status Channel 2 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 2 is not present 1 A hardware service request for channel 2 is present
30 HRS1	Hardware Request Status Channel 1 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 1 is not present 1 A hardware service request for channel 1 is present
31 HRS0	Hardware Request Status Channel 0 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 0 is not present 1 A hardware service request for channel 0 is present

22.3.16 Channel n Priority Register (DMA_DCHPRI*n*)

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMA control register.

Address: 0h base + 100h offset + (1d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	ECP	DPA	GRPPRI		CHPRI			
Write			▨	▨				
Reset	0	0	*	*	*	*	*	*

* Notes:
- CHPRI field: See bit field description.
- GRPPRI field: See bit field description.

DMA_DCHPRI*n* field descriptions

Field	Description
0 ECP	Enable Channel Preemption. 0 Channel n cannot be suspended by a higher priority channel’s service request. 1 Channel n can be temporarily suspended by the service request of a higher priority channel.

Table continues on the next page...

DMA_DCHPRI*n* field descriptions (continued)

Field	Description
1 DPA	Disable Preempt Ability. 0 Channel n can suspend a lower priority channel. 1 Channel n cannot suspend any channel, regardless of channel priority.
2–3 GRPPRI	Channel n Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read-only; writes are ignored. **NOTE:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.
4–7 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled **NOTE:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] = 0b01111.

22.3.17 Channel n Master ID Register (DMA_DCHMID*n*)

The DMA master ID replication registers allow the DMA to use the same protection level and system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master, for example, a core, programs a TCD, its master ID and protection level are captured when the TCDn.CSR control attributes are written. Although the scatter/gather operation can change the contents of TCDn_CSR, that operation does not affect the DCHMID *n* registers.

Address: 0h base + 140h offset + (1d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	EMI	PAL	0		MID			
Write								
Reset	0	0	0	0	0	0	0	0

DMA_DCHMID*n* field descriptions

Field	Description
0 EMI	Enable Master ID replication **NOTE:** If Master ID replication is disabled, the privileged protection level (supervisor mode) for DMA transfers is used.

Table continues on the next page...

DMA_DCHMID*n* field descriptions (continued)

Field	Description
	0 Master ID replication is disabled 1 Master ID replication is enabled
1 PAL	Privileged Access Level **NOTE:** The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCDn.CSR. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MID	Master ID DMA's master ID when channel n is active and master ID replication is enabled. **NOTE:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCDn.CSR.

22.3.18 TCD Source Address (DMA_TCD*n*_SADDR)

Address: 0h base + 1000h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SADDR field descriptions

Field	Description
0–31 SADDR	Source Address Memory address pointing to the source data.

22.3.19 TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Address: 0h base + 1004h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMOD					SSIZE			DMOD					DSIZE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_ATTR field descriptions

Field	Description
0–4 SMOD	Source Address Modulo 0 Source address modulo feature is disabled ≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
5–7 SSIZE	Source data transfer size **NOTE:** Using a Reserved value causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte burst (4 beats of 64 bits) 110 Reserved 111 Reserved
8–12 DMOD	Destination Address Modulo See the SMOD definition
13–15 DSIZE	Destination data transfer size See the SSIZE definition

22.3.20 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Address: 0h base + 1006h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SOFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SOFF field descriptions

Field	Description
0–15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

22.3.21 TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD*n*_NBYTES_MLNO)

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_NBYTES_MLNO field descriptions

Field	Description
0–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

22.3.22 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMLOE	DMLOE	NBYTES													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_NBYTES_MLOFFNO field descriptions

Field	Description
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
2–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

22.3.23 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMLOE	DMLOE	MLOFF													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MLOFF						NBYTES									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR

Table continues on the next page...

DMA_TCD*n*_NBYTES_MLOFFYES field descriptions (continued)

Field	Description
2–21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

22.3.24 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Address: 0h base + 100Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SLAST field descriptions

Field	Description
0–31 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

22.3.25 TCD Destination Address (DMA_TCD*n*_DADDR)

Address: 0h base + 1010h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DADDR field descriptions

Field	Description
0–31 DADDR	Destination Address Memory address pointing to the destination data.

22.3.26 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)

If TCDn_CITER[ELINK] is set, the TCDn_CITER register is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	ELINK		LINKCH					CITER
Write		0						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1 Reserved	This field is reserved.
2–6 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
7–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

22.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

If TCDn_CITER[ELINK] is cleared, the TCDn_CITER register is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	CITER						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

22.3.28 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Address: 0h base + 1016h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DOFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DOFF field descriptions

Field	Description
0–15 DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

22.3.29 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Address: 0h base + 1018h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DLASTSGA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DLASTSGA field descriptions

Field	Description
0–31 DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, otherwise a configuration error is reported.

22.3.30 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	ELINK		LINKCH					BITER
Write		0						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	BITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_BITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1 Reserved	This field is reserved.
2–6 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
7–15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

22.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	BITER						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	BITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

22.3.32 TCD Control and Status (DMA_TCD*n*_CSR)

Address: 0h base + 101Eh offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	BWC			MAJORLINKCH				
Write			0					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CSR field descriptions

Field	Description
0–1 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. **NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00 No eDMA engine stalls. 01 Reserved 10 eDMA engine stalls for 4 cycles after each R/W. 11 eDMA engine stalls for 8 cycles after each R/W.
2 Reserved	This field is reserved.
3–7 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
8 DONE	Channel Done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated. **NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits.
9 ACTIVE	Channel Active This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected.

Table continues on the next page...

DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
11 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
12 DREQ	Disable Request If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero. 0 The channel's ERQ bit is not affected. 1 The channel's ERQ bit is cleared when the major loop is complete.
13 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
14 INTMAJOR	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
15 START	Channel Start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution.

Table continues on the next page...

DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
	0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

22.4 Functional description

The operation of the eDMA is described in the following subsections.

22.4.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

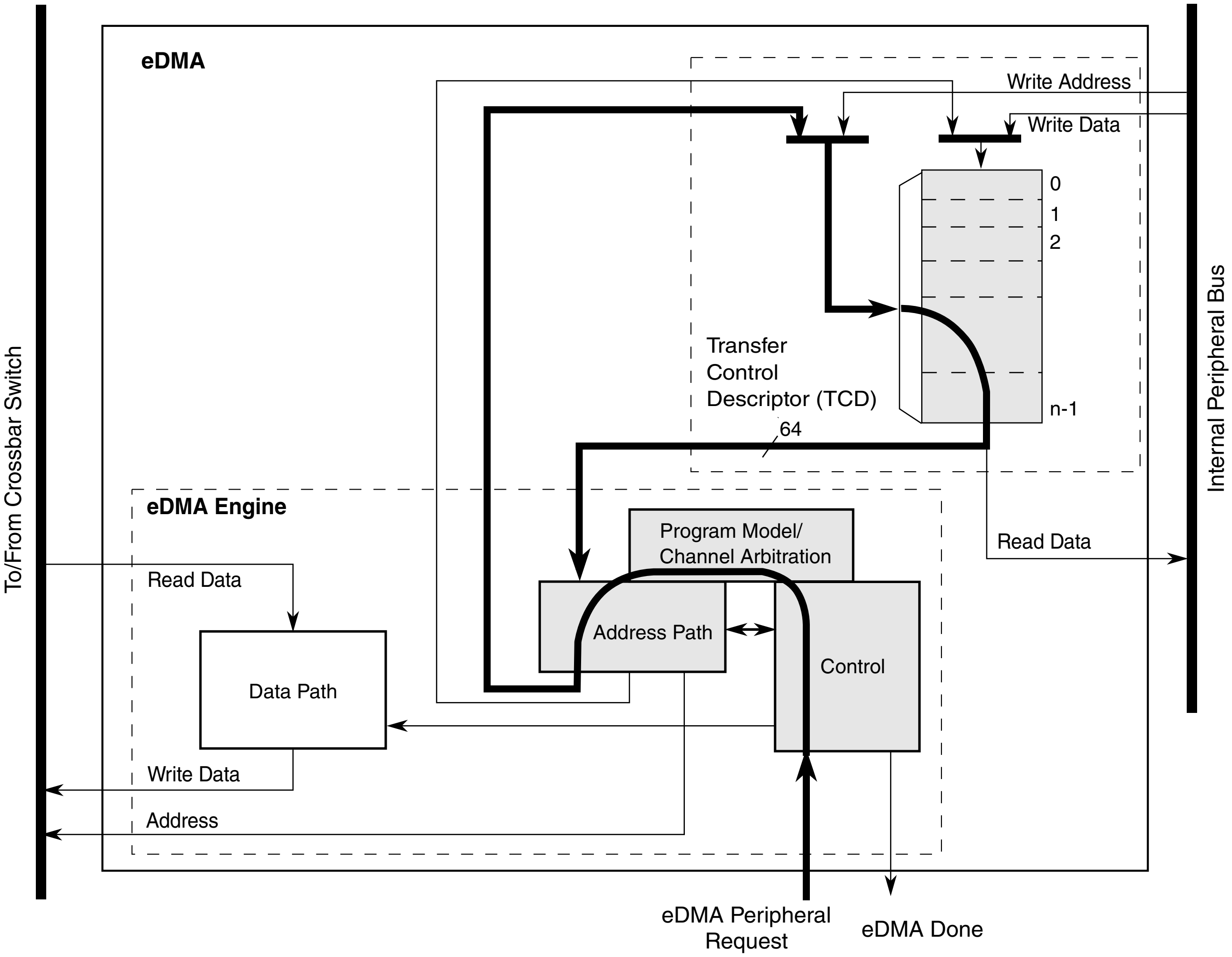


Figure 22-2. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration performs, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

The following diagram illustrates the second part of the basic data flow:

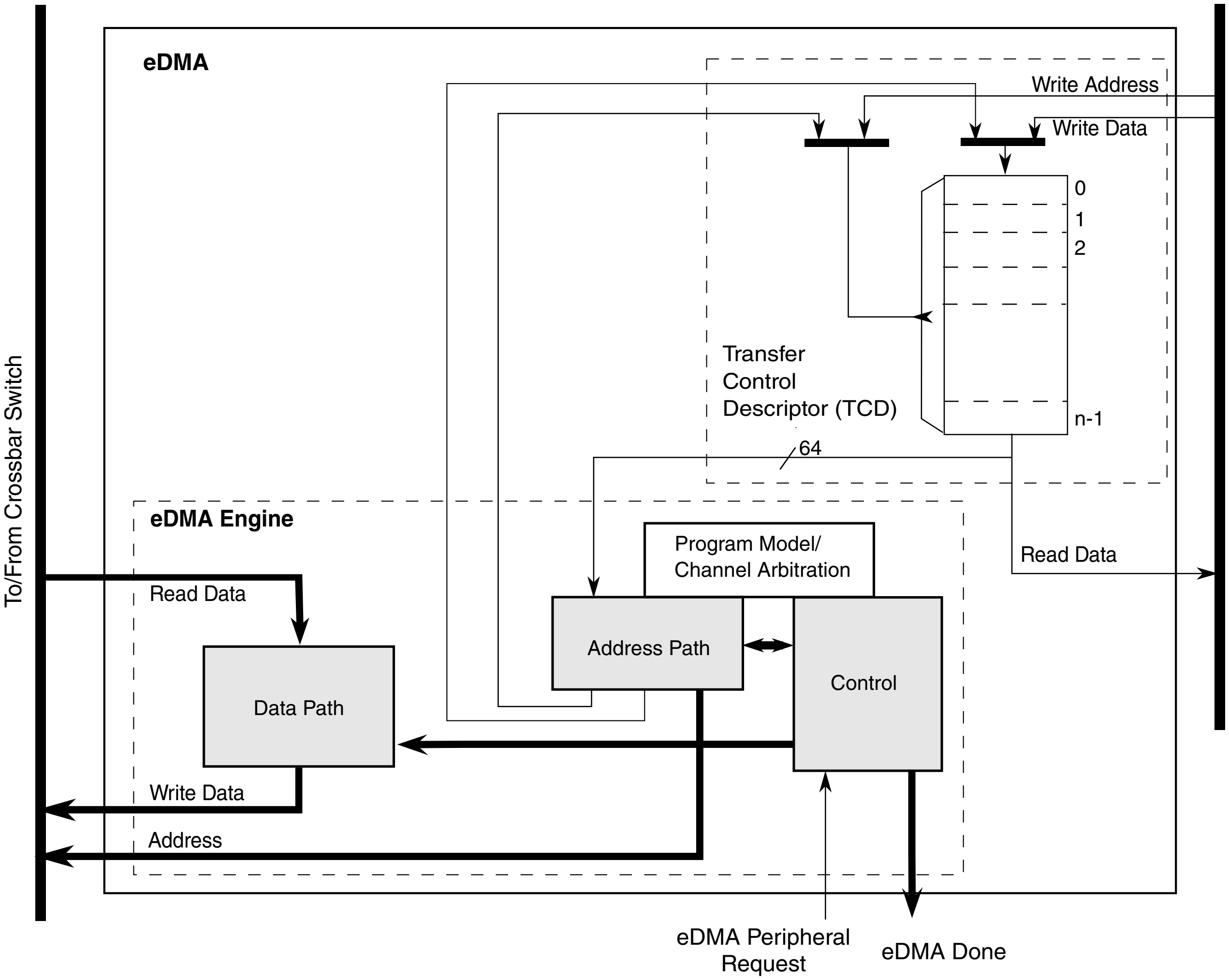


Figure 22-3. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

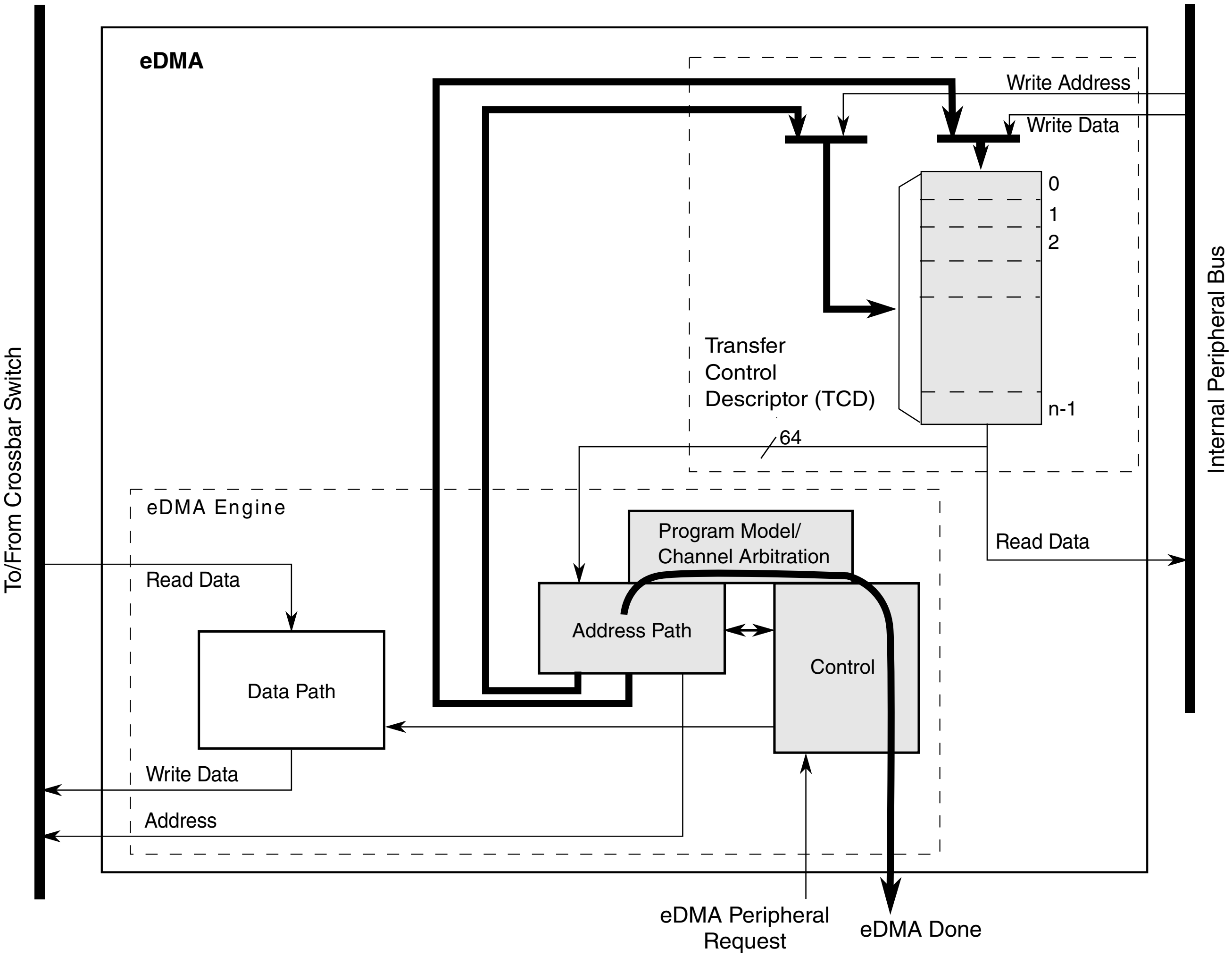


Figure 22-4. eDMA operation, part 3

22.4.2 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.

- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration in undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the Halt bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

> The cancel transfer request allows the user to stop a large data transfer in the event the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must handle the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor since the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

22.4.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

22.4.4 Performance

This section addresses the performance of the eDMA module, focusing on two separate metrics:

- In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces.
- In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more relevant metric. In this environment, the speed of the source and destination address spaces remains important. However, the microarchitecture of the eDMA also factors significantly into the resulting metric.

22.4.4.1 Peak transfer rates

The peak transfer rates for several different source and destination transfers are shown in the following tables. These tables assume:

- Internal SRAM can be accessed with zero wait-states when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states, when viewed from the system bus data phase
- All internal peripheral bus accesses are 32-bits in size

NOTE

All architectures will not meet the assumptions listed above. See the SRAM configuration section for more information.

This table compares peak transfer rates based on different possible system speeds. Specific chips/devices may not support all system speeds listed.

Table 22-4. eDMA peak transfer rates (Mbytes/sec)

System Speed, Width	Internal SRAM-to- Internal SRAM	32 bit internal peripheral bus-to-Internal SRAM	Internal SRAM-to-32 bit internal peripheral bus
66.7 MHz, 64 bit	266.7	66.6	53.3
83.3 MHz, 64 bit	333.3	83.3	66.7
100.0 MHz, 64 bit	400.0	100.0	80.0
133.3 MHz, 64 bit	533.3	133.3	106.7
150.0 MHz, 64 bit	600.0	150.0	120.0

Internal-SRAM-to-internal-SRAM transfers occur at the core's datapath width. For all transfers involving the internal peripheral bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

22.4.4.2 Peak request rates

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, assume that the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel.

The eDMA design supports the following hardware service request sequence. Note that the exact timing from Cycle 7 is a function of the response times for the channel's read and write accesses. In the case of an internal peripheral bus read and internal SRAM write, the combined data phase time is 4 cycles. For an SRAM read and internal peripheral bus write, it is 5 cycles.

Table 22-5. Hardware service request process

Cycle		Description
With internal peripheral bus read and internal SRAM write	With SRAM read and internal peripheral bus write	
1		eDMA peripheral request is asserted.
2		The eDMA peripheral request is registered locally in the eDMA module and qualified. TCD*n*_CSR[START] bit initiated requests start at this point with the registering of the user write to TCD*n* word 7.
3		Channel arbitration begins.
4		Channel arbitration completes. The transfer control descriptor local memory read is initiated.
5–6		The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles
7		The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
8–11	8–12	The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.
12	13	This cycle represents the data phase of the last destination write.
13	14	The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n* word 7 is read and checked for channel linking or scatter/gather requests.
14	15	The appropriate fields in the first part of the TCD*n* are written back into the local memory.
15	16	The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
16	17	The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (4 cycles) and SRAM-to-internal peripheral bus (5 cycles), DMA requests can be processed every 11.5 cycles (4 + (4+5)/2 + 3). This is the time from Cycle 4 to Cycle x +5. The resulting peak request rate, as a function of the system frequency, is shown in the following table.

Table 22-6. eDMA peak request rate (MReq/sec)

System frequency (MHz)	Request rate with zero wait states	Request rate with wait states
66.6	7.4	5.8
83.3	9.2	7.2
100.0	11.1	8.7
133.3	14.8	11.6
150.0	16.6	13.0

A general formula to compute the peak request rate with overlapping requests is:

PEAKreq = freq / [entry + (1 + read_ws) + (1 + write_ws) + exit]

where:

Table 22-7. Peak request formula operands

Operand	Description
PEAKreq	Peak request rate
freq	System frequency
entry	Channel startup (4 cycles)
read_ws	Wait states seen during the system bus read data phase
write_ws	Wait states seen during the system bus write data phase
exit	Channel shutdown (3 cycles)

22.4.4.3 eDMA performance example

Consider a system with the following characteristics:

- Internal SRAM can be accessed with one wait-state when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states viewed from the system bus data phase
- System operates at 150 MHz

For an SRAM to internal peripheral bus transfer, PEAKreq = 150 MHz / [4 + (1 + 1) + (1 + 3) + 3] cycles = 11.5 Mreq/sec

For an internal peripheral bus to SRAM transfer,

PEAKreq = 150 MHz / [4 + (1 + 2) + (1 + 1) + 3] cycles = 12.5 Mreq/sec

Assuming an even distribution of the two transfer types, the average peak request rate would be:

PEAKreq = (11.5 Mreq/sec + 12.5 Mreq/sec) / 2 = 12.0 Mreq/sec

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start where no channel is executing and eDMA is idle are:

- 11 cycles for a software, that is, a TCD*n*_CSR[START] bit, request
- 12 cycles for a hardware, that is, an eDMA peripheral request signal, request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlaps the previous executing channel.

Note

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

22.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

22.5.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.

3. Enable error interrupts in the EEI register if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQH and ERQL registers.
6. Request channel service via either:
   - Software: setting the TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 22-8. TCD Control and Status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

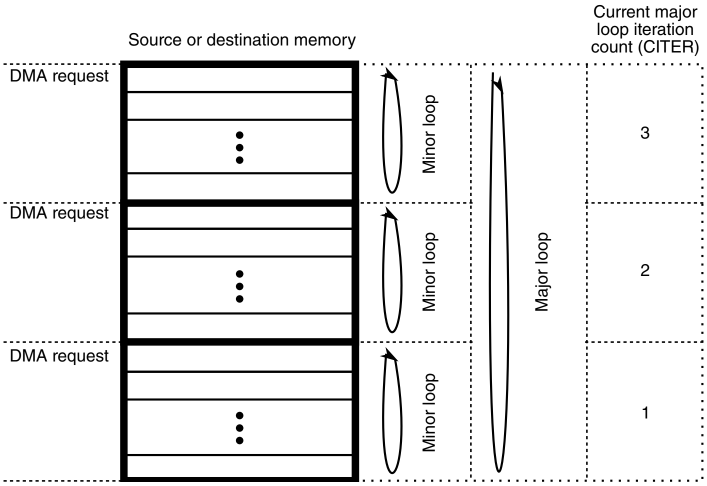


Figure 22-5. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

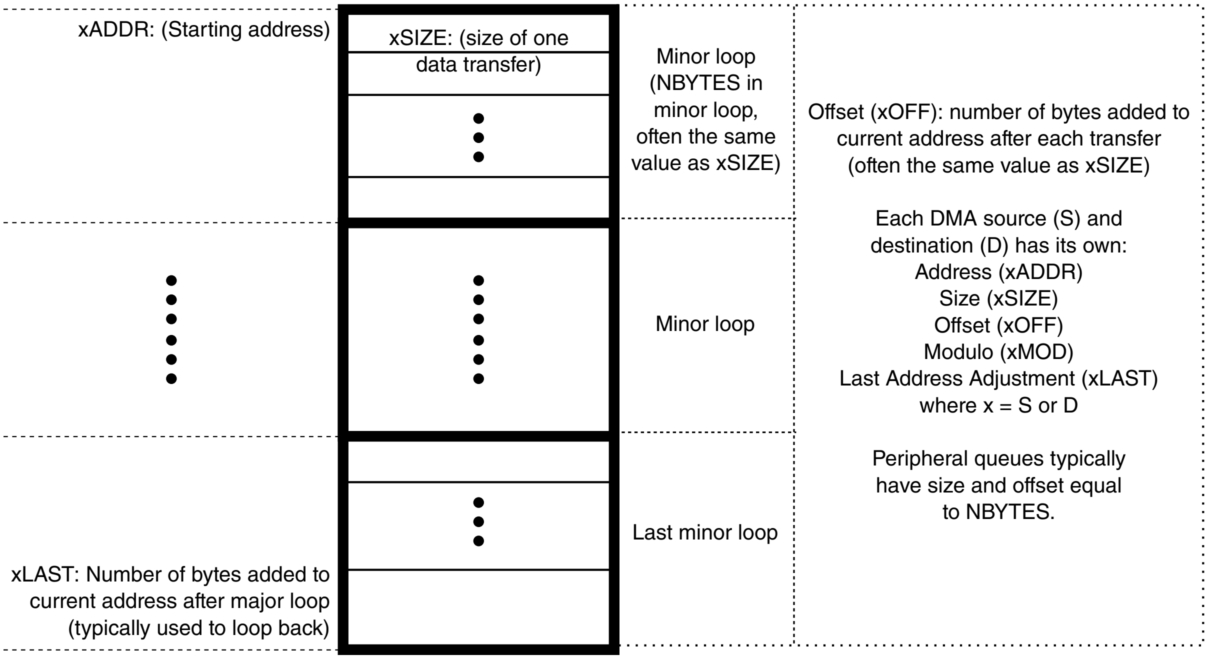

xADDR: (Starting address)
xSIZE: (size of one data transfer)
Minor loop (NBYTES in minor loop, often the same value as xSIZE)
Offset (xOFF): number of bytes added to current address after each transfer (often the same value as xSIZE)
Each DMA source (S) and destination (D) has its own:
Address (xADDR)
Size (xSIZE)
Offset (xOFF)
Modulo (xMOD)
Last Address Adjustment (xLAST)
where x = S or D
Peripheral queues typically have size and offset equal to NBYTES.
Minor loop
Last minor loop
xLAST: Number of bytes added to current address after major loop (typically used to loop back)

Figure 22-6. Memory array terms

22.5.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group 1 is the highest priority and all channels are unique in that group.
3. Group 0 is the next highest priority and has two channels with the same priority level.
4. If Group 1 has any service requests, those requests will be executed.
5. After all of Group 1 requests have completed, Group 0 will be the next active group.
6. If Group 0 has a service request, then an undefined channel in Group 0 will be selected and a channel priority error will occur.
7. This repeats until the all of Group 0 requests have been removed or a higher priority Group 1 request comes in.

In this sequence, for item 2, the eDMA acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel/group priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

22.5.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

22.5.3.1 Fixed group arbitration, Fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so that the channels within one group use "fixed" priorities, and that group is assigned the highest "fixed" priority of all groups, that group can take all the bandwidth of the eDMA controller. That is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

22.5.3.2 Fixed group arbitration, Round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, Fixed channel arbitration, but all the channels in the highest priority group will be serviced. Service latency will be short on the highest priority group, but could potentially be very much longer as the group priority decreases.

22.5.4 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

22.5.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32-bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32-bits to location 0x200C → last iteration of the minor loop → major loop complete.

6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

22.5.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, eDMA peripheral, request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32-bits to location 0x2008 → third iteration of the minor loop.

   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32-bits to location 0x200C → last iteration of the minor loop.

6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
    a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
    b. Write 32-bits to location 0x2010 → first iteration of the minor loop.
    c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
    d. Write 32-bits to location 0x2014 → second iteration of the minor loop.
    e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
    f. Write 32-bits to location 0x2018 → third iteration of the minor loop.
    g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
    h. Write 32-bits to location 0x201C → last iteration of the minor loop → major loop complete.
14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.

16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

22.5.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 22-9. Modulo example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

22.5.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

22.5.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware, that is, peripheral, initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

22.5.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and

DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

22.5.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

22.5.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit

2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done→ set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

> The TCD*n*_CITER[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

Table 22-10. Channel Linking Parameters

Desired Link Behavior	TCD Control Field Name	Description
Link at end of Minor Loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of Major Loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

22.5.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

22.5.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode,
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

22.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution (see the diagram in TCD structure). This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write 1 to the TCD.major.e_link bit.
2. Read back the TCD.major.e_link bit.
3. Test the TCD.major.e_link request status:
    - If TCD.major.e_link = 1, the dynamic link attempt was successful.
    - If TCD.major.e_link = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

22.5.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources.When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

22.5.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD indentification (ID).

1. When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCD.d_req bit.

Should a dynamic scatter/gather attempt fail, setting the TCD.d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

3. Write the TCD.dlast_sga field with the scatter/gather address.
4. Write 1b to the TCD.e_sg bit.
5. Read back the 16 bit TCD control/status field.
6. Test the TCD.e_sg request status and TCD.major.linkch value:

   If e_sg = 1b, the dynamic link attempt was successful.

   If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

22.5.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD indentification (ID).

1. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

2. Write theTCD.dlast_sga field with the scatter/gather address.
3. Write 1b to the TCD.e_sg bit.
4. Read back the TCD.e_sg bit.
5. Test the TCD.e_sg request status:

   If e_sg = 1b, the dynamic link attempt was successful.

   If e_sg = 0b, read the 32 bit TCD dlast_sga field.

   If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring).

If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

Chapter 23
Direct Memory Access Multiplexer (DMAMUX)

23.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

23.1.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in the following figure.

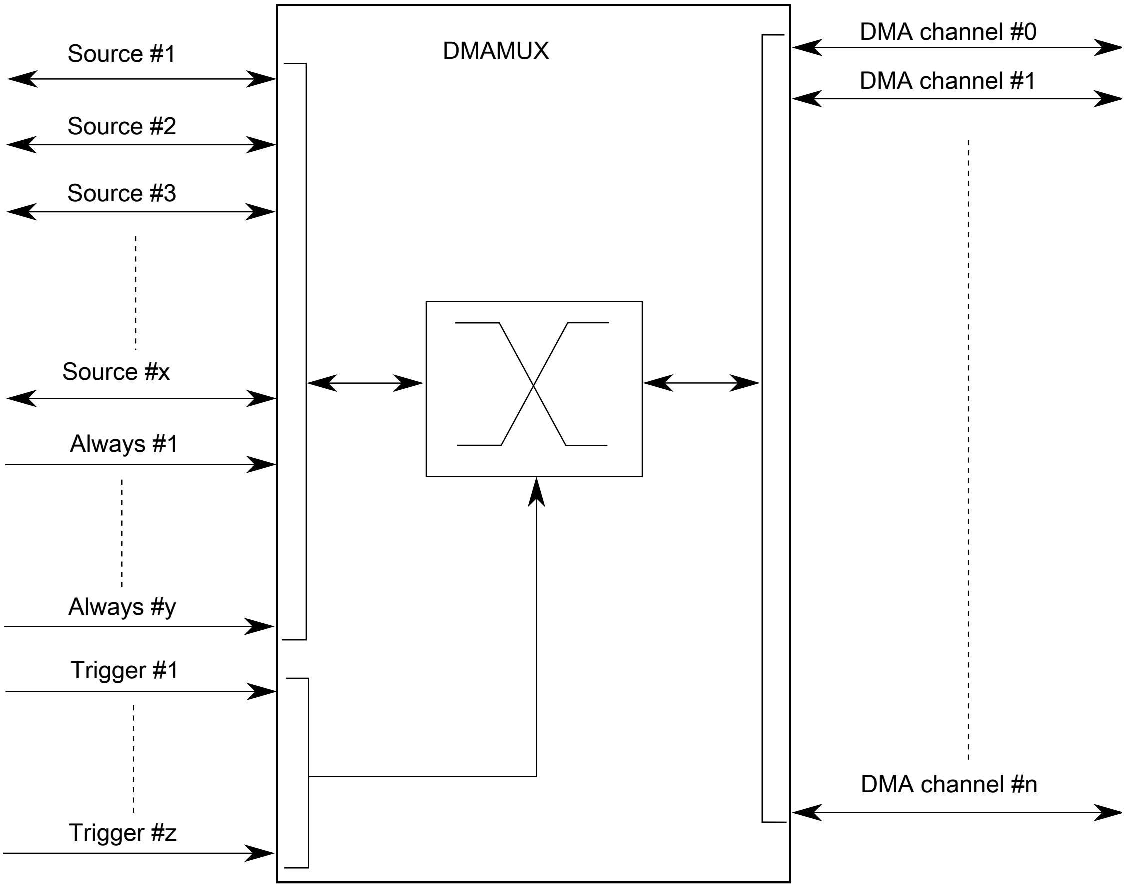


Figure 23-1. DMAMUX block diagram

23.1.2 Features

The DMAMUX module provides these features:

- Up to 27 peripheral slots and up to six always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channel routers.
    - The first four channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

23.1.3 Modes of operation

The following operating modes are available:

- Disabled mode

In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically.

  Configuration of the period is done in the registers of the periodic interrupt timer (PIT). This mode is available only for channels 0–3.

23.2 External signal description

The DMAMUX has no external pins.

23.3 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

DMAMUX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Channel Configuration register (DMAMUX_CHCFG0)	8	R/W	00h	23.3.1/742
1	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	23.3.1/742
2	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	23.3.1/742
3	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	23.3.1/742
4	Channel Configuration register (DMAMUX_CHCFG4)	8	R/W	00h	23.3.1/742
5	Channel Configuration register (DMAMUX_CHCFG5)	8	R/W	00h	23.3.1/742
6	Channel Configuration register (DMAMUX_CHCFG6)	8	R/W	00h	23.3.1/742
7	Channel Configuration register (DMAMUX_CHCFG7)	8	R/W	00h	23.3.1/742
8	Channel Configuration register (DMAMUX_CHCFG8)	8	R/W	00h	23.3.1/742

Table continues on the next page...

DMAMUX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9	Channel Configuration register (DMAMUX_CHCFG9)	8	R/W	00h	23.3.1/742
A	Channel Configuration register (DMAMUX_CHCFG10)	8	R/W	00h	23.3.1/742
B	Channel Configuration register (DMAMUX_CHCFG11)	8	R/W	00h	23.3.1/742
C	Channel Configuration register (DMAMUX_CHCFG12)	8	R/W	00h	23.3.1/742
D	Channel Configuration register (DMAMUX_CHCFG13)	8	R/W	00h	23.3.1/742
E	Channel Configuration register (DMAMUX_CHCFG14)	8	R/W	00h	23.3.1/742
F	Channel Configuration register (DMAMUX_CHCFG15)	8	R/W	00h	23.3.1/742

23.3.1 Channel Configuration register (DMAMUX_CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

> Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).
>
> Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7
Read Write	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

DMAMUX_CHCFG*n* field descriptions

Field	Description
0 ENBL	DMA Channel Enable Enables the DMA channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMAMux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 1 DMA channel is enabled
1 TRIG	DMA Channel Trigger Enable Enables the periodic trigger capability for the triggered DMA channel.

Table continues on the next page...

DMAMUX_CHCFG*n* field descriptions (continued)

Field	Description
	0 Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) 1 Triggering is enabled. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode.
2–7 SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

23.4 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

23.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 4 channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention.

The trigger is generated by the periodic interrupt timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. See the section on periodic interrupt timer for more information on this topic.

Note

Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

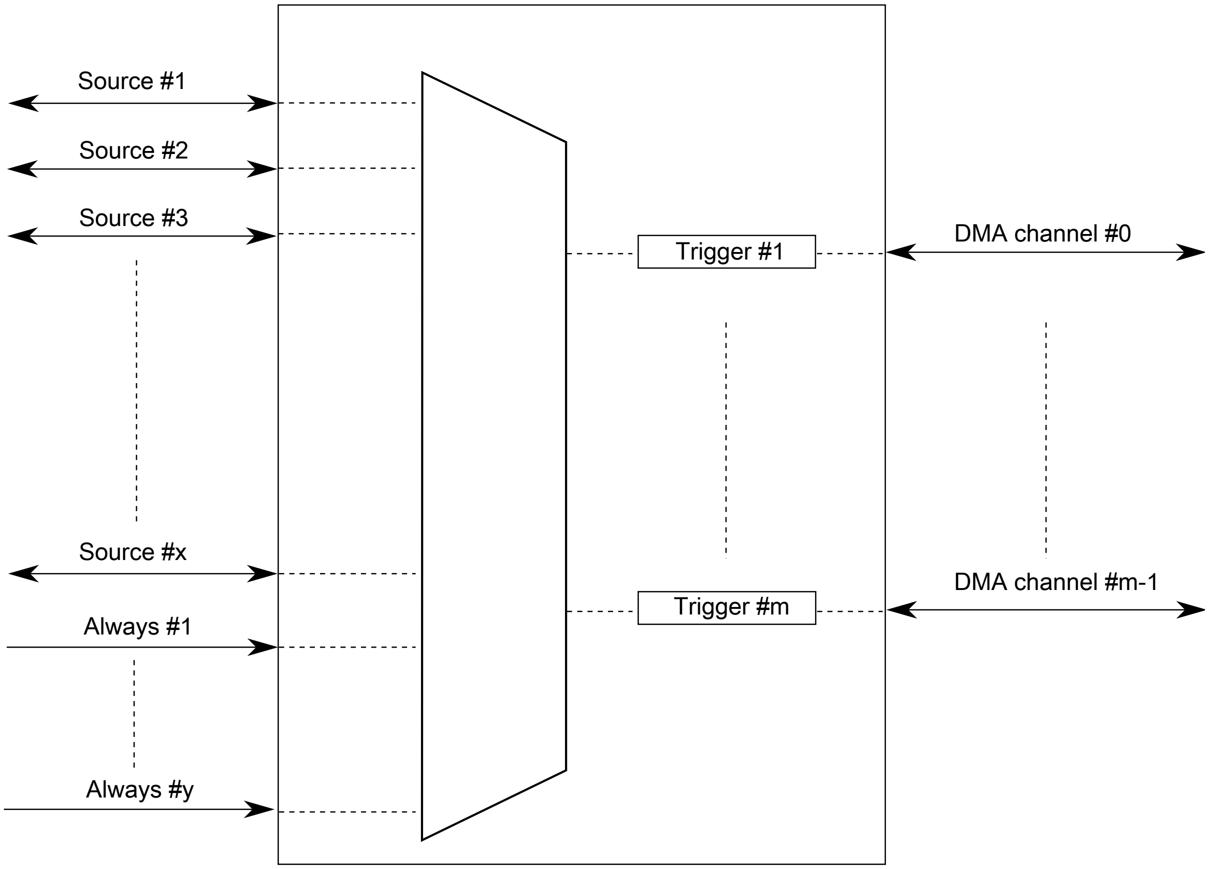


Figure 23-2. DMAMUX triggered channels

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.

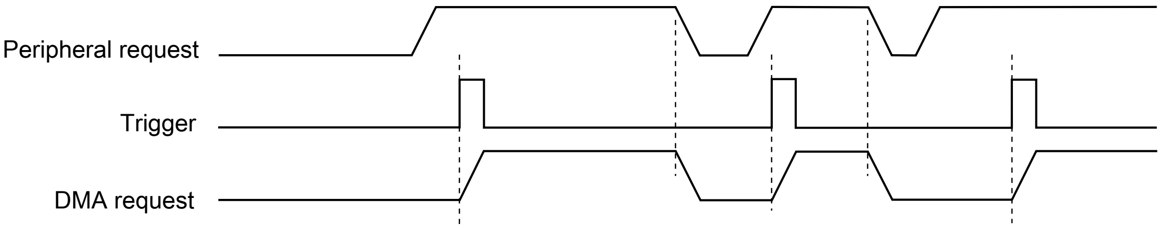


Figure 23-3. DMAMUX channel triggering: normal operation

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.

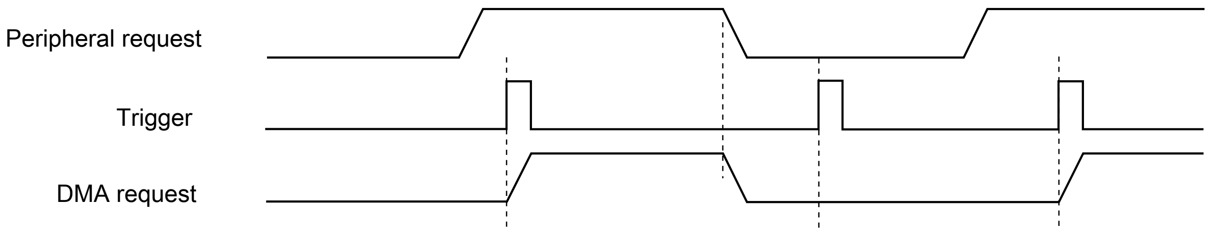


Figure 23-4. DMAMUX channel triggering: ignored trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 μs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

  By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

23.4.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in Modes of operation.

23.4.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are six additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

23.5 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

23.5.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

23.5.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00 to CHCFG1.

2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG1.

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

To enable a source, without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

NOTE

> The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0x00 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;
*CHCFG1 = 0x85;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8.
3. Write 0x87 to CHCFG8. (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
```

```
In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;
*CHCFG8 = 0x87;
```

Chapter 24
Error Injection Module (EIM)

24.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Error Injection module is mainly used for diagnostic purposes. It provides a method for diagnostic coverage of the peripheral memories. See the chip-specific EIM information to determine which peripheral memories are supported by this method.

24.1.1 Overview

The Error Injection Module (EIM) provides support for inducing single-bit and multi-bit inversions on read data when accessing peripheral RAMs. Injecting faults on memory accesses can be used to exercise the SEC-DED ECC function of the related system.

NOTE

> The following diagram shows an example EIM implementation with a 64-bit read data bus and an 8-bit checkbit bus.

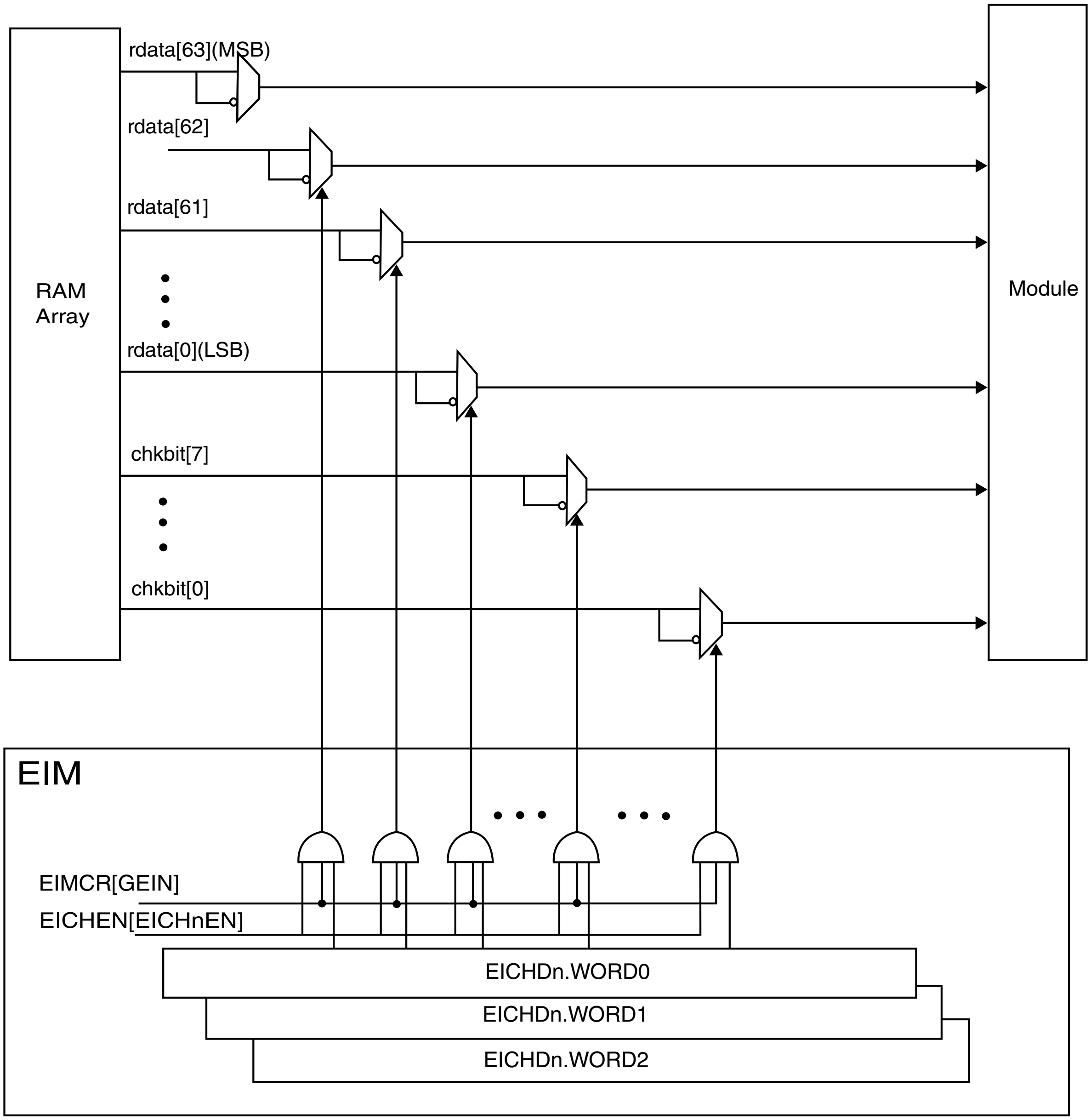


Figure 24-1. EIM functional block diagram (64-bit read data bus and 8-bit checkbit bus)

24.1.2 Features

The EIM includes these features:

- Supports 1 error injection channel
- Protection against accidental enable and reconfiguration error injection function via two-stage enable mechanism

24.2 Memory map and register definition

The EIM provides an IPS programming model mapped to an on-platform peripheral slot.

Programming model access

All system bus masters can access the programming model:

- Only in supervisor mode
- Using only 32-bit (word) accesses

Any of the following attempted references to the programming model generates an IPS error termination:

- In user mode
- Using non-32-bit access sizes
- To undefined (reserved) addresses

Attempted updates to the programming model while the EIM is in the midst of an operation will result in non-deterministic behavior.

Error injection channel descriptor: function and structure

Each error injection channel descriptor:

- Specifies a mask that defines which bits of the read data and checkbit bus from target RAM are inverted on a read access.
- Consists of a 128-bit (16-byte) structure, composed of four 32-bit words, in the EIM programming model.
    - Each descriptor consists of Error Injection Channel Descriptor Word0-1 (EICHD*n*_WORD0-1) and, depending on the implementation, Word2 (EICHD*n*_WORD2).
    - When Word2 is unused, the corresponding word in the 16-byte structure is unused.
    - The fourth word in the 16-byte structure is always unused.

EIM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Error Injection Module Configuration Register (EIM_EIMCR)	32	R/W	0000_0000h	24.2.1/754
4	Error Injection Channel Enable register (EIM_EICHEN)	32	R/W	0000_0000h	24.2.2/755
100	Error Injection Channel Descriptor, Word0 (EIM_EICHD0_WORD0)	32	R/W	0000_0000h	24.2.3/756
104	Error Injection Channel Descriptor, Word1 (EIM_EICHD0_WORD1)	32	R/W	0000_0000h	24.2.4/757
108	Error Injection Channel Descriptor, Word2 (EIM_EICHD0_WORD2)	32	R/W	0000_0000h	24.2.5/757

24.2.1 Error Injection Module Configuration Register (EIM_EIMCR)

The EIM Configuration Register is used to globally enable/disable the error injection function.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															GEIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EIMCR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GEIEN	Global Error Injection Enable This bit globally enables or disables the error injection function of the EIM. This field is initialized by hardware reset. 0 Disabled 1 Enabled

24.2.2 Error Injection Channel Enable register (EIM_EICHEN)

Each field of the Error Injection Channel Enable register (EICHEN) is used to enable or disable the corresponding error injection channel.

NOTE

> To enable an error injection channel, the Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EICH0EN	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHEN field descriptions

Field	Description
0 EICH0EN	Error Injection Channel 0 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. This field is cleared by power-on reset and unaffected by other types of system reset. 0 Error injection is disabled on Error Injection Channel 0 1 Error injection is enabled on Error Injection Channel 0
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

24.2.3 Error Injection Channel Descriptor, Word0 (EIM_EICHD*n*_WORD0)

The first word of the Error Injection Channel Descriptor defines a left-justified mask field (CHKBIT_MASK) whose width is up to 8 bits. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

Address: 0h base + 100h offset + (256d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHKBIT_MASK								0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHD*n*_WORD0 field descriptions

Field	Description
0–7 CHKBIT_MASK	Checkbit Mask This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. The width of the field for the channel is defined in the chip-specific EIM information. 0 The corresponding bit of the checkbit bus remains unmodified. 1 The corresponding bit of the checkbit bus is inverted.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

24.2.4 Error Injection Channel Descriptor, Word1 (EIM_EICHD*n*_WORD1)

The second word of the Error Injection Channel Descriptor defines the bits in the mask corresponding to bytes 0–3 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

Address: 0h base + 104h offset + (256d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B0_3DATA_MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHD*n*_WORD1 field descriptions

Field	Description
0–31 B0_3DATA_ MASK	Data Mask Bytes 0-3 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 0-3 of the read data bus, see the chip-specific EIM information. 0 The corresponding bit of bytes 0-3 on the read data bus remains unmodified 1 The corresponding bit of bytes 0-3 on the read data bus is inverted.

24.2.5 Error Injection Channel Descriptor, Word2 (EIM_EICHD*n*_WORD2)

The third word of the Error Injection Channel Descriptor, when present, defines the bits in the mask corresponding to bytes 4–7 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

Address: 0h base + 108h offset + (256d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B4_7DATA_MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHD*n*_WORD2 field descriptions

Field	Description
0–31 B4_7DATA_ MASK	Data Mask Bytes 4-7 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 4-7 of the read data bus, see the chip-specific EIM information. 0 The corresponding bit of bytes 4-7 on the read data bus remains unmodified. 1 The corresponding bit of bytes 4-7 on the read data bus is inverted.

24.3 Functional description

The EIM provides protection against accidental enabling and reconfiguration of the error injection function by enforcing a two-stage enablement mechanism. To properly enable the error injection mechanism for a channel, execute both of the following:

- Assert the EICHEN[EICH*n*EN] field, where *n* denotes the channel number.
- Assert EIMCR[GEIEN].

NOTE

When the use case for a channel requires writing any EICHD*n*_WORD register, write the EICHD*n*_WORD register before executing the two-stage enablement mechanism. A successful write to any EICHD*n*_WORD register clears the corresponding EICHEN[EICH*n*EN] field.

The EIM supports 1 error injection channel. Each channel:

- is assigned to a single memory array interface.
- intercepts the assigned memory read data bus and checkbit bus and injects errors by inverting the value transmitted for selected bits on each bus line.

On a memory read access, the applicable EICHD*n*_WORD registers define which bits of the read data and/or checkbit bus to invert.

Figure 24-1 depicts the interception and override of a 64-bit read data bus and an 8-bit checkbit data bus for an example memory array.

Chapter 25
Dual PLL Digital Interface (PLLDIG)

25.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Dual PLL system composed of PLL0 and PLL1 analog blocks and the digital interface (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1.

A key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by PLL1.

25.2 Block Diagram

Please refer to the “Clocking chapter” of this *Reference Manual* to see the block diagram.

25.3 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for decreasing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.

- Programmable frequency modulation on PLL1.
- Lock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
    - The loss of lock indication is sent to the FCCU via the system glue logic.

25.4 Modes of operation

The operating mode of the PLLs is determined by the value of PLL*n*CR[CLKCFG]. PLLDIG operates in normal mode with a reference clock and PLL enabled.

NOTE

Mode changes are controlled by configuring the MC_ME Mode Configuration registers (MC_ME_<*mode*>_MC).

Normal mode is defined as the mode where the clocks are driven by the PLLs. When using a crystal for the reference clock, reset should remain asserted until the oscillator has stabilized.

25.4.1 Normal mode with reference, PLL0 or both PLLs enabled

In normal mode, PLL0 receives an input clock from the reference which can be prescaled by the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0 PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, and modulation enable are programmed by configuring the PLLFM register.

NOTE

While powering down the PLLs and if PLL0:PHI1 is used as the source for PLL1, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Configuring the appropriate MC_ME_<*mode*>_MC register to start PLL1 should only be done after PLL0 has achieved lock.

25.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs. The table below shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

PLLDIG memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)	32	R/W	See section	25.5.1/762
4	PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)	32	R/W	See section	25.5.2/764
8	PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)	32	R/W	See section	25.5.3/766
20	PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)	32	R/W	See section	25.5.4/768
24	PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)	32	R/W	See section	25.5.5/770
28	PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)	32	R/W	See section	25.5.6/771
2C	PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)	32	R/W	See section	25.5.7/773
30	PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)	32	R/W	0000_0000h	25.5.8/775

25.5.1 PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)

The PLL0CR is shown in the following table.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					Reserved	CLKCFG		Reserved	Reserved	Reserved	Reserved	LOLIE	Reserved	LOCIE	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0	0*

* Notes:
- This field can be written at any time, but writes are ignored. Read returns previously written value.
- LOLIE field: See the "Clocking" chapter for details on configuring this field.

PLLDIG_PLL0CR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 CLKCFG	Clock Configuration This field indicates the operational state of PLL.

Table continues on the next page...

PLLDIG_PLL0CR field descriptions (continued)

Field	Description
	When PLLs are externally powered down the CLKCFG field changes to 00b. **NOTE:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1. **NOTE:** CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_< *mode* >_MC register (see the "Mode Entry Module (MC_ME)" chapter and Clock configuration for details). 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 Reserved	This field is reserved.
30 LOCIE	Loss-of-clock interrupt request. The LOCIE bit enables a loss-of-clock interrupt request when PLL0SR[LOCF] = 1. If PLL0SR[LOCF] = 0 or PLL0CR[LOCIE] = 0, the interrupt request is ignored. 0 Ignore loss-of-lock. Reset not asserted. 1 Assert reset on loss-of-lock when operating in normal mode.
31 Reserved	This field is reserved.

25.5.2 PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								Reserved	Reserved	Reserved	Reserved	LOLF	LOCK	Reserved	Reserved
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL0SR field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset.

Table continues on the next page...

PLLDIG_PLL0SR field descriptions (continued)

Field	Description
	0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

25.5.3 PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider. PLL0DV can be modified at anytime, but the changes become effective only after the PLL is disabled, then reenabled. If these fields are changed without powering down the PLL, the PLL will lose lock and generate either a reset or interrupt based on which is enabled. The reduced frequency divider bitfields (RFDPHI, RFDPHI1) can be modified at anytime, but the changes only become effective when PLL0 is disabled, then reenabled.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	RFDPHI1				Reserved					RFDPHI					
W																
Reset	0	*	*	*	*	0	0	0	0	0	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	PREDIV			Reserved					MFD						
W																
Reset	0	*	*	*	0	0	0	0	0	*	*	*	*	*	*	*

* Notes:
- MFD field: See Clocking chapter for reset value
- PREDIV field: See Clocking chapter for reset value
- RFDPHI field: See Clocking chapter for reset value

- RFDPHI1 field: See Clocking chapter for reset value

PLLDIG_PLL0DV field descriptions

Field	Description
0 Reserved	This field is reserved.
1–4 RFDPHI1	PHI1 reduced frequency divider. This 4-bit field determines the VCO clock post divider for driving the PHI1 output clock. RFDPHI1 has a range of 4..15 (4h..Fh). All other values are reserved.
5–9 Reserved	This field is reserved.
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. RFDPHI has a range of 1..63 (1h..3Fh). All other values are reserved.
16 Reserved	This field is reserved.
17–19 PREDIV	Input clock predivider. This field controls the value of the divider on the input clock. The output of the predivider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 011 Divide by 3 100 Divide by 4 101 Divide by 5 110 Divide by 6 111 Divide by 7
20–24 Reserved	This field is reserved.
25–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 8...127 (8h...7Fh). All other values are reserved (see Clock configuration for details).

25.5.4 PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)

The PLL1CR is shown in the following table. It contains the operational state of PLL1, and is used to enable/disable PLL specific interrupts.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					Reserved	CLKCFG		Reserved	Reserved	Reserved	Reserved	LOLIE	Reserved	LOCIE	Reserved
W																
Reset	0	0	0	0	0	0*	0	0	0*	0*	0*	0*	0*	0*	0	0*

* Notes:
- This field can be written at any time, but writes are ignored. Read returns previously written value.
- LOLIE field: Please refer to device "Clocking" chapter for details on configuring this field.

PLLDIG_PLL1CR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 CLKCFG	Clock Configuration This field indicates the operational state of the PLL. When the PLLs are externally powered down the CLKCFG field changes to 00b. **NOTE:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.

Table continues on the next page...

PLLDIG_PLL1CR field descriptions (continued)

Field	Description
	NOTE: CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_< *mode* >_MC register (see the "Mode Entry Module (MC_ME)" chapter and Clock configuration for details). 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 Reserved	This field is reserved.
30 LOCIE	Loss-of-clock interrupt request. The LOCIE bit enables a loss-of-clock interrupt request when PLL1SR[LOCF] = 1. If PLL1SR[LOCF] = 0 or PLL1CR[LOCIE] = 0, the interrupt request is ignored. 0 Ignore loss-of-lock. Reset not asserted. 1 Assert reset on loss-of-lock when operating in normal mode.
31 Reserved	This field is reserved.

25.5.5 PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)

PLL1SR contains status flags showing the current state of the PLL.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved						0	0	Reserved	Reserved	Reserved	Reserved	LOLF	LOCK	Reserved	Reserved
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL1SR field descriptions

Field	Description
0–21 Reserved	This field is reserved.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.

Table continues on the next page...

PLLDIG_PLL1SR field descriptions (continued)

Field	Description
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

25.5.6 PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)

The values of the reduced frequency divider (RFDPHI) and loop multiplication factor divider (MFD) can be modified at anytime, but the new values only become effective after the PLL is disabled, then reenabled.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved										RFDPHI						Reserved					Reserved				MFD						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*

* Notes:
- MFD field: See Clocking chapter for reset value
- RFDPHI field: See Clocking chapter for reset value

PLLDIG_PLL1DV field descriptions

Field	Description
0–9 Reserved	This field is reserved.

Table continues on the next page...

PLLDIG_PLL1DV field descriptions (continued)

Field	Description
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. RFDPHI has a range of 1..63 (1h..3Fh). All other values are reserved.
16–20 Reserved	This field is reserved.
21–24 Reserved	This field is reserved.
25–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 16...34 (10h...22h). All other values are reserved.

25.5.7 PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)

The PLL1FM register controls enables frequency modulation, and provides controls for modulation mode selection, modulation depth and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). The PLL would then be required to be power cycled to regain normal functionality.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	MODEN	MODSEL	MODPRD												
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	INCSTP														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL1FM field descriptions

Field	Description
0 Reserved	This field is reserved.

Table continues on the next page...

PLLDIG_PLL1FM field descriptions (continued)

Field	Description
1 MODEN	Modulation enable. This bit enables the frequency modulation. 0 Frequency modulation disabled. 1 Frequency modulation enabled.
2 MODSEL	Modulation selection. This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Centre Spread Modulation - Modulation centred around nominal frequency. 1 Down Spread Modulation - Modulation spread below nominal frequency.
3–15 MODPRD	Modulation period. MODPRD is the modulation period variable derived from the formulas shown in Frequency modulation.
16 Reserved	This field is reserved.
17–31 INCSTP	Increment step. This field is the INCSTP variable derived from the formulas shown in Frequency modulation.

25.5.8 PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (PLL*n*CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	FDEN	Reserved												Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved				FRCDIV											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL1FD field descriptions

Field	Description
0 Reserved	This field is reserved.
1 FDEN	Fractional Divide Enable This bit enables the fractional divider in the loop divider for PLL1.

Table continues on the next page...

PLLDIG_PLL1FD field descriptions (continued)

Field	Description
	0 Fractional divide disabled. 1 Fractional divide enabled.
2–13 Reserved	This field is reserved.
14–15 Reserved	Reserved. This field is reserved. **Important:** This field must be written 00b.
16–19 Reserved	This field is reserved.
20–31 FRCDIV	Fractional divide input. When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the mean VCO clock frequency is given by the equation in the section Clock configuration.

25.6 Register classification for safety requirements

This module is classified as ViMo (Vital Modules) for functional safety. Hence, the registers of this module have been classified as shown in the Functional Safety chapter.

25.7 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

25.7.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for the PLL s are discussed in detail in the *Data Sheet*.

25.7.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV, and PLL1FD registers, and calculated according to the following equations:

$$f_{\text{pll0\_phi}} = f_{\text{pll0\_ref}} \times \left(\frac{\text{PLL0DV[MFD]}}{\text{PLL0DV[PREDIV]} \times \text{PLL0DV[RFDPHI]}} \right)$$

Equation 1. PLL0 PHI Output Frequency

$$f_{\text{pll0\_phi1}} = f_{\text{pll0\_ref}} \times \left(\frac{\text{PLL0DV[MFD]}}{\text{PLL0DV[PREDIV]} \times \text{PLL0DV[RFDPHI1]}} \right)$$

Equation 2. PLL0 PHI1 Output Frequency

$$f_{\text{pll1\_phi}} = f_{\text{pll1\_ref}} \times \left(\frac{\text{PLL1DV[MFD]} + \frac{\text{PLL1FD[FRCDIV]}}{2^{12}} + \frac{1}{2^{13}}}{2 \times \text{PLL1DV[RFDPHI]}} \right)$$

Equation 3. PLL1 PHI Output Frequency

The relationship between the VCO frequency (f_{VCO}) and the output frequency of the PLLs is determined by the configuration of the PLL1DV, PLL1FD, and PLL0DV registers, according to the following equations:

$$f_{\text{pll0\_VCO}} = \frac{f_{\text{pll0\_ref}} \times \text{PLL0DV[MFD]} \times 2}{\text{PLL0DV[PREDIV]}}$$

Equation 4. PLL0 VCO Frequency

$$f_{\text{pll1\_VCO}} = f_{\text{pll1\_ref}} \times \left(\text{PLL1DV[MFD]} + \frac{\text{PLL1FD[FRCDIV]}}{2^{12}} + \frac{1}{2^{13}} \right)$$

Equation 5. PLL1 VCO Frequency

NOTE

$f_{\text{pll0\_phi1}}$ is the reference clock generated by PLL0 for PLL1

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification of PLL0 and PLL1. Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of the PLL*n*DV registers are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLL*n*CR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

Note

> The PLL must be powered down and powered up by configuring, then executing, a mode change in the MC_ME_<*mode*>_MC before PLL0DV[PREDIV], PLL0DV[MFD], PLL1DV[MFD], or the input clocks are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in Initialization information.

25.7.3 Loss of lock

The PLL digital interface registers provide the flexibility to select whether to generate an interrupt, assert system reset, or do nothing in the event that the PLL loses lock according to the available PLL*n*CR options.

Loss of lock indication is only generated when the PLL is operating in normal mode.

The lock indication from the analog PLL is synchronized and stored in the status register. When the analog PLL loses lock or regains lock, it will be immediately indicated in the status register (after a synchronization time).

A pair of counters monitor the reference and feedback clocks to determine when the system has acquired frequency lock. Once the PLL has locked, the counters continue to monitor the reference and feedback clocks and report whether the PLL has lost lock. The lock status is indicated in the status register.

25.7.4 Frequency modulation

Frequency modulation uses a triangular profile as shown in Figure 25-1. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

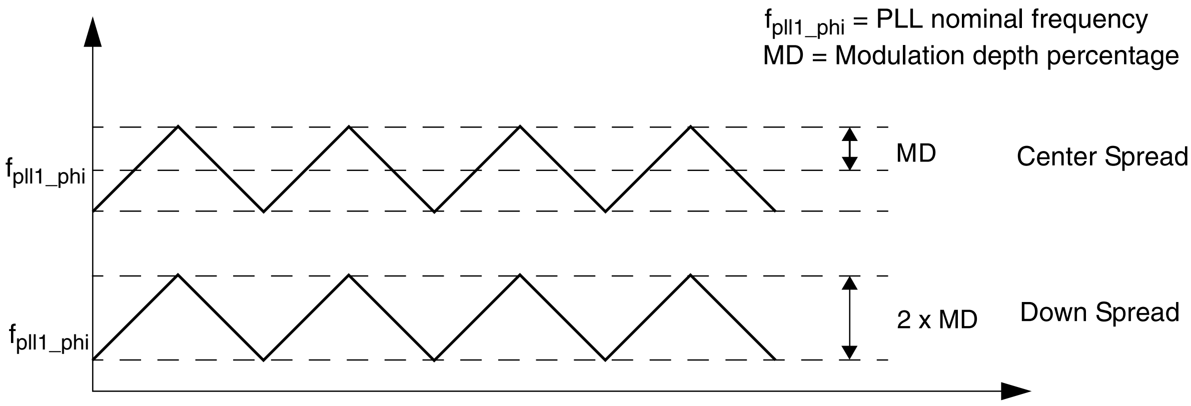

f_{pll1_phi} = PLL nominal frequency
MD = Modulation depth percentage
f_{pll1_phi}
MD
Center Spread
f_{pll1_phi}
2 x MD
Down Spread

Figure 25-1. Triangular frequency modulation

> **NOTE**
>
> The device maximum operating frequency includes the frequency modulation. If center modulation is used, the f_{sys} must be below f_{max} by MD percentage such that $f_{max} = f_{sys} \times (1 + (MD\% / 100))$.

The following equations define how to calculate PLL1FM[MODPRD] and PLL1FM[INCSTP] based on the output frequency of the feedback divider (f_{ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD).

$$\mathrm{PLL1FM[MODPRD]} = \mathrm{round}\left(\frac{f_{\mathrm{pll1\_ref}}}{4 \times f_{\mathrm{mod}}}\right)$$

Equation 6. PLL1FM[MODPRD] Calculation

The equation to determine PLL1FM[INCSTP] is shown in Equation 7 on page 779.

$$\mathrm{PLL1FM[INCSTP]} = \mathrm{round}\left(\frac{(2^{15} - 1) \times \mathrm{MD} \times \mathrm{PLL1DV[MFD]}}{100 \times 5 \times \mathrm{PLL1FM[MODPRD]}}\right)$$

Equation 7. PLL1FM[INCSTP] Calculation

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

$$(\mathrm{PLL1FM[MODPRD]} \times \mathrm{PLL1FM[INCSTP]}) < 2^{15}$$

Equation 8. PLL1FM[MODPRD] and PLL1FM[INCSTP] Restriction

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is shown in Equation 9 on page 779.

$$\text{ModulationDepth} = \text{round}\left(\frac{\text{PLL1FM[MODPRD]} \times \text{PLLIFM[INCSTP]} \times 100 \times 5}{(2^{15} - 1) \times \text{PLL1DV[MFD]}}\right)$$

Equation 9. Modulation Depth

FM parameters should only be changed, and FM enabled, after PLL0 has obtained lock. The sequence for reprogramming the FM is:

1. Power down PLL1 by writing MC_ME_<*mode*>_MC[PLL1ON] = 0, followed by a mode change.
2. Write a mode change to MC_ME_<*mode*>_MC[PLL1ON].
3. Program the PLL1FM while PLL1 is powered down.
4. Power up the PLL1 by writing MC_ME_<*mode*>_MC[PLL1ON] = 1, followed by a mode change.
5. Write a mode change to MC_ME_<*mode*>_MC[PLL1ON].

25.8 Initialization information

Coming out of reset PLL0 and PLL1 are disabled per the DRUN mode configuration register, MC_ME_DRUN_MC. The Dual PLLDIG initialization procedure is described in the "Clocking" chapter.

NOTE

See the "Mode Entry Module (MC_ME)" chapter in this *Reference Manual* more details.

Chapter 26
Clock Monitor Unit (CMU)

26.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock source CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and detects if the monitored clock frequency leaves an upper or lower frequency boundary

NOTE

See the "Clocking" chapter for chip-specific sources used by the CMU.

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take the necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate an event if CLKMN1 is greater than a high frequency boundary or less than a low frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of a clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

NOTE

See the "Clocking" chapter for the number of CMU instances on this chip.

26.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN ÷ $2^{CSR[RCDIV]}$ clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to CLKMT0_RMN ÷ 4.
- Event generation for various failures detected inside monitoring unit.

26.2 Block diagram

The block diagram of the CMU module(s) is shown in the “Clocking" chapter of this Reference Manual.

26.3 Signals

The table below describes the signals on the boundary of the CMU (in alphabetical order).

Table 26-1. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference — Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1 — Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.

Table continues on the next page...

Table 26-1. Signal description (continued)

Signal	I/O	Description
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference — Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

NOTE

See the "Clocking" chapter for device specific clock sources of each CMU.

26.4 Register description and memory map

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in the following table.

NOTE

See "Clocking" chapter for register and field availability details.

CMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CMU Control Status Register (CMU_CSR)	32	R/W	See section	26.4.1/784
4	CMU Frequency Display Register (CMU_FDR)	32	R	0000_0000h	26.4.2/785
8	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	32	R/W	0000_0FFFh	26.4.3/786
C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	32	R/W	0000_0000h	26.4.4/786
10	CMU Interrupt Status Register (CMU_ISR)	32	w1c	See section	26.4.5/787
18	CMU Measurement Duration Register (CMU_MDR)	32	R/W	0000_0000h	26.4.6/789

26.4.1 CMU Control Status Register (CMU_CSR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								SFM	0						
W																
Reset	0	0	0	0	0	0	0	0	0*	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CKSEL1		0					RCDIV		CME
W																
Reset	0	0	0	0	0	0	0*	0*	0	0	0	0	0	0*	0*	0

* Notes:

- RCDIV field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- CKSEL1 field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- SFM field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.

CMU_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SFM	Start frequency measure. The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CKSEL1	Frequency measure clock selection bit. CKSEL1 selects the clock to be measured by the frequency meter. This only affects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected 01 Reserved 10 Reserved 11 CLKMT0_RMN is selected
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 RCDIV	CLKMT0_RMN division factor.

Table continues on the next page...

CMU_CSR field descriptions (continued)

Field	Description
	These bits specify the CLKMT0_RMN division factor. The output clock frequency is $f_{CLKMT0\_RMN} \div 2^{CMU\_CSR[RCDIV]}$. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. 00 CLKMT0_RMN ÷ 1 (No division) 01 CLKMT0_RMN ÷ 2 10 CLKMT0_RMN ÷ 4 11 CLKMT0_RMN ÷ 8
31 CME	CLKMN1 monitor enable. 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

26.4.2 CMU Frequency Display Register (CMU_FDR)

The CMU_FDR is used to determine the measured frequency of:

- CLKMT0_RMN

with respect to the reference clock CLKMN0_RMT.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												FD																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMU_FDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 FD	Measured frequency bits. This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMN0\_RMT} \times \text{CMU\_MDR[MD]}) \div \text{CMU\_FDR[FD]}$

26.4.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_HFREFR register.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																				HFREF											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

CMU_HFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 HFREF	High Frequency reference value. These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

26.4.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_LFREFR register.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																				LFREF											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMU_LFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CMU_LFREFR field descriptions (continued)

Field	Description
20–31 LFREF	Low Frequency reference value. These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

26.4.5 CMU Interrupt Status Register (CMU_ISR)

NOTE

All flags in the CMU_ISR are set asynchronously. This register must be read only after an interrupt(safe mode entry) is triggered by the CMU. Otherwise, a read access on this register may fetch an incorrect value (see "Clocking" chapter for interrupt operation) .

NOTE

Before entering low-power stop modes, all clock frequency measurements in the CMU should be disabled. Failing to do so may result in spurious interrupts.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved												Reserved	FHHI	FLLI	OLRI
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*

* Notes:

- OLRI field: Not all CMU blocks will utilize this feature. See the "Clocking" chapter for specific CMU implementation details.

CMU_ISR field descriptions

Field	Description
0–27 Reserved	This field is reserved.
28 Reserved	This field is reserved.
29 FHHI	CLKMN1 frequency higher than high reference event status. This bit is set by hardware when CLKMN1 frequency becomes higher than the high frequency reference value determined by CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred
30 FLLI	CLKMN1 frequency less than low reference event status. This bit is set by hardware when CLKMN1 frequency becomes lower than the low frequency reference value determined by CMU_LFREFR[LFREF], and CLKMN1 is 'ON' as signaled by the MC_ME. Software clears this field by writing a '1'. 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$event status. This bit is set by hardware when the f_{CLKMN0_RMT} is less than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$frequency and CLKMN0_RMT is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. **NOTE:** When attempting to enter STOP mode with the chip running on the PLL, erroneous OLRI triggers may occur. To avoid this the XOSC should be stopped before entering STOP mode. 0 No OLR event 1 OLR event occurred

26.4.6 CMU Measurement Duration Register (CMU_MDR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												MD																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMU_MDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1.

26.5 Functional description

26.5.1 Frequency meter

The purpose of the frequency meter is to evaluate the deviation from the nominal metered source (such as CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of the metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the "Clocking" chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR[FD] as shown in the following equation:

$$f_{\mathrm{sel}} = f_{\mathrm{CLKMN0\_RMT}} \times \frac{\mathrm{CMU\_MDR[MD]}}{\mathrm{CMU\_FDR[FD]}}$$

26.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than the frequency of CLKMT0_RMN ÷ $2^{\text{CMU\_CSR[RCDIV]}}$ and CLKMN0_RMT is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

Note

$f_{\text{CLKMN0\_RMT}}$ must be greater than $f_{\text{CLKMT0\_RMN}} \div 2^{\text{CMU\_CSR[RCDIV]}}$ by at least 0.5 MHz in order to guarantee correct $f_{\text{CLKMN0\_RMT}}$ monitoring.

26.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If f_{CLKMN1} is greater than the reference value determined by fields CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

If f_{CLKMN1} is less than a reference value determined by the bits CMU_LFREFR[LFREF] and the CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note

An example of determining the $\text{HFREF}_{\text{Actual}}$ is as follows. Assume a $f_{\text{CLKMT0\_RMN}}$ = 16 MHz with a accuracy of +/-5%. In order to monitor a f_{CLKMN1} = 200 MHz, the ideal $\text{HFREF}_{\text{Ideal}}$ = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration ($\text{HFREF}_{\text{Actual}} = (\text{HFREF}_{\text{Ideal}} \div 0.95)$.

The actual LFREF value will be 762 when accuracy is taken into consideration ($LFREF_{Actual} = LFREF_{Ideal} \div 1.05$).

Chapter 27
Clock Generation Module (MC_CGM)

27.1 Introduction

27.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the chip blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for details). A set of MC_CGM registers controls the clock dividers that are used for divided system and peripheral clock generation. The memory spaces of system and peripheral clock sources that have addressable memory spaces are accessed through the MC_CGM memory space.

The following figure is the MC_CGM block diagram.

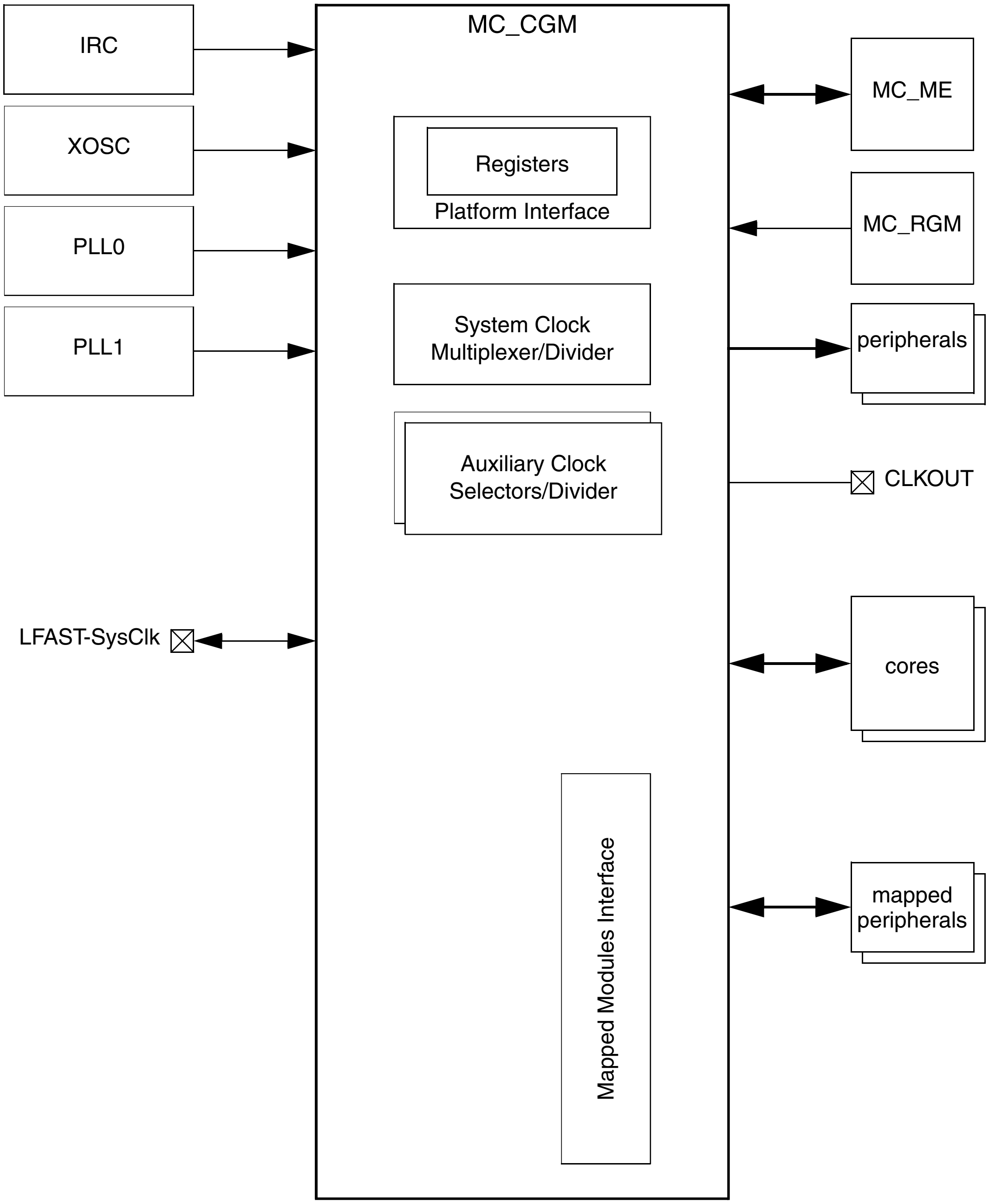


Figure 27-1. MC_CGM Block Diagram

27.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control

- performs progressive system clock frequency change depending on MC_ME mode configuration
- contains a set of registers to control clock dividers for divided clock generation
- supports multiple clock sources and maps their address spaces to its memory map
- guarantees glitch-less clock transitions when changing the system clock selection
- supports 8, 16, and 32-bit wide read/write accesses

27.2 External Signal Description

The MC_CGM delivers output clocks to the CLKOUT pin for off-chip use and/or observation.

The LFAST-SysClk pin can be either driven by the MC_CGM or used as an input to source the LFAST PLL clock.

27.3 Memory Map and Register Definition

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0704, as a half-word at address offset 0x0706, or as a byte at address offset 0x0707.

MC_CGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
700	PCS Switch Duration Register (MC_CGM_PCS_SDUR)	8	R/W	See section	27.3.1/797
704	PCS Divider Change Register 1 (MC_CGM_PCS_DIVC1)	32	R/W	See section	27.3.2/797
708	PCS Divider End Register 1 (MC_CGM_PCS_DIVE1)	32	R/W	See section	27.3.3/798
70C	PCS Divider Start Register 1 (MC_CGM_PCS_DIVS1)	32	R/W	See section	27.3.4/799
710	PCS Divider Change Register 2 (MC_CGM_PCS_DIVC2)	32	R/W	See section	27.3.5/800
714	PCS Divider End Register 2 (MC_CGM_PCS_DIVE2)	32	R/W	See section	27.3.6/800

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
718	PCS Divider Start Register 2 (MC_CGM_PCS_DIVS2)	32	R/W	See section	27.3.7/801
728	PCS Divider Change Register 4 (MC_CGM_PCS_DIVC4)	32	R/W	See section	27.3.8/802
72C	PCS Divider End Register 4 (MC_CGM_PCS_DIVE4)	32	R/W	See section	27.3.9/803
730	PCS Divider Start Register 4 (MC_CGM_PCS_DIVS4)	32	R/W	See section	27.3.10/ 803
7E4	System Clock Select Status Register (MC_CGM_SC_SS)	32	R	See section	27.3.11/ 804
7E8	System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)	32	R/W	8001_0000h	27.3.12/ 806
800	Auxiliary Clock 0 Select Control Register (MC_CGM_AC0_SC)	32	R/W	0000_0000h	27.3.13/ 806
804	Auxiliary Clock 0 Select Status Register (MC_CGM_AC0_SS)	32	R/W	0000_0000h	27.3.14/ 807
808	Auxiliary Clock 0 Divider 0 Configuration Register (MC_CGM_AC0_DC0)	32	R/W	0000_0000h	27.3.15/ 808
80C	Auxiliary Clock 0 Divider 1 Configuration Register (MC_CGM_AC0_DC1)	32	R/W	0000_0000h	27.3.16/ 809
810	Auxiliary Clock 0 Divider 2 Configuration Register (MC_CGM_AC0_DC2)	32	R/W	0000_0000h	27.3.17/ 809
828	Auxiliary Clock 1 Divider 0 Configuration Register (MC_CGM_AC1_DC0)	32	R/W	0000_0000h	27.3.18/ 810
82C	Auxiliary Clock 1 Divider 1 Configuration Register (MC_CGM_AC1_DC1)	32	R/W	0000_0000h	27.3.19/ 811
848	Auxiliary Clock 2 Divider 0 Configuration Register (MC_CGM_AC2_DC0)	32	R/W	0000_0000h	27.3.20/ 812
860	Auxiliary Clock 3 Select Control Register (MC_CGM_AC3_SC)	32	R	0000_0000h	27.3.21/ 813
864	Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS)	32	R	0000_0000h	27.3.22/ 814
880	Auxiliary Clock 4 Select Control Register (MC_CGM_AC4_SC)	32	R/W	0100_0000h	27.3.23/ 815
884	Auxiliary Clock 4 Select Status Register (MC_CGM_AC4_SS)	32	R	0100_0000h	27.3.24/ 816
8A0	Auxiliary Clock 5 Select Control Register (MC_CGM_AC5_SC)	32	R/W	0100_0000h	27.3.25/ 816
8A4	Auxiliary Clock 5 Select Status Register (MC_CGM_AC5_SS)	32	R/W	0100_0000h	27.3.26/ 817
8A8	Auxiliary Clock 5 Divider 0 Configuration Register (MC_CGM_AC5_DC0)	32	R/W	0000_0000h	27.3.27/ 818
8C0	Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)	32	R/W	0000_0000h	27.3.28/ 818
8C4	Auxiliary Clock 6 Select Status Register (MC_CGM_AC6_SS)	32	R	0000_0000h	27.3.29/ 819
8C8	Auxiliary Clock 6 Divider 0 Configuration Register (MC_CGM_AC6_DC0)	32	R/W	0000_0000h	27.3.30/ 820

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
940	Auxiliary Clock 10 Select Control Register (MC_CGM_AC10_SC)	32	R/W	0000_0000h	27.3.31/ 821
944	Auxiliary Clock 10 Select Status Register (MC_CGM_AC10_SS)	32	R	0000_0000h	27.3.32/ 822
948	Auxiliary Clock 10 Divider 0 Configuration Register (MC_CGM_AC10_DC0)	32	R/W	0000_0000h	27.3.33/ 822
960	Auxiliary Clock 11 Select Control Register (MC_CGM_AC11_SC)	32	R/W	0000_0000h	27.3.34/ 823
964	Auxiliary Clock 11 Select Status Register (MC_CGM_AC11_SS)	32	R	0000_0000h	27.3.35/ 824
968	Auxiliary Clock 11 Divider 0 Configuration Register (MC_CGM_AC11_DC0)	32	R/W	0000_0000h	27.3.36/ 825

27.3.1 PCS Switch Duration Register (MC_CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See Progressive System Clock Switching for details on how to set this value.

This register is reset only on a power-on reset.

Address: 0h base + 700h offset = 700h

Bit	0	1	2	3	4	5	6	7
Read Write	SDUR							
Reset	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- Reset by POR only

MC_CGM_PCS_SDUR field descriptions

Field	Description
0–7 SDUR	Switch Duration This value defines the duration of one PCS clock switch step in terms of 16 MHz IRCOSC cycles.

27.3.2 PCS Divider Change Register 1 (MC_CGM_PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the 8-40 MHz XOSC on ramp-up and ramp-down, respectively. See Progressive System Clock Switching for details on how to set these values.

This register is reset only on a power-on reset.

NOTE

> Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 704h offset = 704h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset by POR only

MC_CGM_PCS_DIVC1 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to the 8-40 MHz XOSC.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the 8-40 MHz XOSC.

27.3.3 PCS Divider End Register 1 (MC_CGM_PCS_DIVE1)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the 8-40 MHz XOSC on ramp-down. See Progressive System Clock Switching for details on how to set this value.

This register is reset only on a power-on reset.

NOTE

> Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 708h offset = 708h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*

* Notes:

- Reset by POR only

MC_CGM_PCS_DIVE1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the 8-40 MHz XOSC.

27.3.4 PCS Divider Start Register 1 (MC_CGM_PCS_DIVS1)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the 8-40 MHz XOSC on ramp-up. Register *n* corresponds to system clock source *n* . See Progressive System Clock Switching for details on how to set these values.

This register is reset only on a power-on reset.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 70Ch offset = 70Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*

* Notes:

- Reset by POR only

MC_CGM_PCS_DIVS1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the 8-40 MHz XOSC.

27.3.5 PCS Divider Change Register 2 (MC_CGM_PCS_DIVC2)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL0 PHI on ramp-up and ramp-down, respectively. See Progressive System Clock Switching for details on how to set these values.

This register is reset only on a power-on reset.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 710h offset = 710h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset by POR only

MC_CGM_PCS_DIVC2 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value — This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate — This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL0 PHI.

27.3.6 PCS Divider End Register 2 (MC_CGM_PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL0 PHI on ramp-down. See Progressive System Clock Switching for details on how to set this value.

This register is reset only on a power-on reset.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 714h offset = 714h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*

* Notes:

- Reset by POR only

MC_CGM_PCS_DIVE2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the PLL0 PHI.

27.3.7 PCS Divider Start Register 2 (MC_CGM_PCS_DIVS2)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL0 PHI on ramp-up. See Progressive System Clock Switching for details on how to set this value.

This register is reset only on a power-on reset.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 718h offset = 718h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*

* Notes:

- Reset by POR only

MC_CGM_PCS_DIVS2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.

27.3.8 PCS Divider Change Register 4 (MC_CGM_PCS_DIVC4)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL1 PHI on ramp-up and ramp-down, respectively. See Progressive System Clock Switching for details on how to set these values.

This register is reset only on a power-on reset.

NOTE

> Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 728h offset = 728h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- Reset by POR only

MC_CGM_PCS_DIVC4 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value - This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate - This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL1 PHI.

27.3.9 PCS Divider End Register 4 (MC_CGM_PCS_DIVE4)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL1 PHI on ramp-down. See Progressive System Clock Switching for details on how to set this value.

This register is reset only on a power-on reset.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 72Ch offset = 72Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*

* Notes:

- Reset by POR only

MC_CGM_PCS_DIVE4 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the PLL1 PHI.

27.3.10 PCS Divider Start Register 4 (MC_CGM_PCS_DIVS4)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL1 PHI on ramp-up. See Progressive System Clock Switching for details on how to set this value.

This register is reset only on a power-on reset.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 730h offset = 730h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	1*	1*	1*	1*	0*	0*	1*	1*	1*

* Notes:

- Reset by POR only

MC_CGM_PCS_DIVS4 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.

27.3.11 System Clock Select Status Register (MC_CGM_SC_SS)

This register provides the current system clock source selection.

Address: 0h base + 7E4h offset = 7E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				SELSTAT				0				SWTRG			SWIP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- SWTRG field: The value of the SWTRG field is “000” after a power-on reset and “100” after all other resets.

MC_CGM_SC_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	System Clock Source Selection Status - This value indicates the current source for the system clock.

Table continues on the next page...

MC_CGM_SC_SS field descriptions (continued)

Field	Description
	NOTE: If the current system clock source is the IRCOSC as reported in the MC_CGM System Clock Select Status Register (CGM_SC_SS[SELSTAT] = 0b0000) and Switch Trigger Cause shows the cause for the latest clock switch as a successful Mode Enter mode change (CGM_SC_SS[SWTRG] = 0b001) then the CGM_SC_SS[SWTRG] will incorrectly continue to show the cause for the latest clock switch as MC_ME succeeded after a SAFE mode request is generated. If a second SAFE mode request is generated then the CGM_SC_SS[SWTRG] switches to report the correct status value. If the CGM_SC_SS[SELSTAT] shows the system clock as IRCOSC, then software should check the Current Mode field of Mode Entry Global Status register (MC_ME_GS[S_CURRENT_MODE]) and the Safe mode Interrupt of Mode Entry Interrupt Status Register (MC_ME_IS[I_SAFE]) to establish the cause of the switch. 0000 16 MHz IRCOSC 0001 8-40 MHz XOSC 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–14 SWTRG	Switch Trigger cause - This value indicates the cause for the latest clock source switch. 000 initial switch to 16 MHz IRCOSC due to power-on reset 001 switch after request from MC_ME succeeded 010 switch after request from MC_ME failed due inactive target clock 011 switch after request from MC_ME failed due inactive current clock 100 switch to 16 MHz IRCOSC due to SAFE mode request or reset succeeded 101 switch to 16 MHz IRCOSC due to SAFE mode request or reset succeeded, but current clock source was inactive 110 reserved 111 reserved
15 SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.12 System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)

This register controls system clock divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

WARNING

Use only odd DIV values (i.e., division factor of 2, 4, 6, 8, etc.). Even values will cause incorrect device behavior.

Address: 0h base + 7E8h offset = 7E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC0 field descriptions

Field	Description
0 DE	Divider 0 Enable **NOTE:** Always write 0b1 to the DE bit when writing to any of the MC_CGM_SC_DCn registers. 0 Disable system clock divider 0 1 Enable system clock divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider 1 Division Value — The resultant AIPS clock will have a period 'DIV + 1' times that of the system clock. If DE is set to '0' (divider 1 is disabled), any write access to the DIV field is ignored and the AIPS clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.13 Auxiliary Clock 0 Select Control Register (MC_CGM_AC0_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 0 divider 0: motor control clock
- divided by auxiliary clock 0 divider 1: SGEN clock
- divided by auxiliary clock 0 divider 2: ADC clock

NOTE

PLL1 is routed on N15P devices into the AUX Clock Selector 0. It is recommended not to enable Frequency modulation on PLL1 when it is being used as a clock source for AUX Clock Selector 0.

See Figure 27-5 for details.

Address: 0h base + 800h offset = 800h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	Auxiliary Clock 0 Source Selection Control — Selects the source for auxiliary clock 0. 000 16 MHz IRCOSC 001 8-40 MHz XOSC 010 PLL0 PHI 100 PLL1 PHI Others Reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.14 Auxiliary Clock 0 Select Status Register (MC_CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.s

Address: 0h base + 804h offset = 804h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	Auxiliary Clock 0 Source Selection Status — This value indicates the current source for auxiliary clock 0. 000 16 MHz IRCOSC 001 8-40 MHz XOSC 010 PLL0 PHI 100 PLL1 PHI Others Reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.15 Auxiliary Clock 0 Divider 0 Configuration Register (MC_CGM_AC0_DC0)

This register controls auxiliary clock 0 divider 0.

NOTE

> Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 808h offset = 808h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 0 1 Enable auxiliary clock 0 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider Division Value — The resultant motor control clock will have a period ’DIV + 1’ times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the motor control clock remains disabled.

Table continues on the next page...

MC_CGM_AC0_DC0 field descriptions (continued)

Field	Description
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.16 Auxiliary Clock 0 Divider 1 Configuration Register (MC_CGM_AC0_DC1)

This register controls auxiliary clock 0 divider 1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 80Ch offset = 80Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0										DIV				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 1 1 Enable auxiliary clock 0 divider 1
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DIV	Divider Division Value — The resultant SGEN clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the SGEN clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.17 Auxiliary Clock 0 Divider 2 Configuration Register (MC_CGM_AC0_DC2)

This register controls auxiliary clock 0 divider 2.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 810h offset = 810h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0										DIV				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC2 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 2 1 Enable auxiliary clock 0 divider 2
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DIV	Divider Division Value — The resultant ADC clock will have a period ’DIV + 1’ times that of auxiliary clock 0. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the ADC clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.18 Auxiliary Clock 1 Divider 0 Configuration Register (MC_CGM_AC1_DC0)

This register controls auxiliary clock 1 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 828h offset = 828h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0										DIV				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC1_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 1 divider 0 1 Enable auxiliary clock 1 divider 0
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DIV	Divider Division Value — The resultant FlexRay clock will have a period ’DIV + 1’ times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the FlexRay clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.19 Auxiliary Clock 1 Divider 1 Configuration Register (MC_CGM_AC1_DC1)

This register controls auxiliary clock 1 divider 1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 82Ch offset = 82Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC1_DC1 field descriptions

Field	Description
0 DE	Divider Enable

Table continues on the next page...

MC_CGM_AC1_DC1 field descriptions (continued)

Field	Description
	0 Disable auxiliary clock 1 divider 1 1 Enable auxiliary clock 1 divider 1
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value — The resultant SENT clock will have a period 'DIV + 1' times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SENT clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.20 Auxiliary Clock 2 Divider 0 Configuration Register (MC_CGM_AC2_DC0)

This register controls auxiliary clock 2 divider 0.

NOTE

> Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 848h offset = 848h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC2_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 2 divider 0 1 Enable auxiliary clock 2 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider Division Value — The resultant CAN clock will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CAN clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.21 Auxiliary Clock 3 Select Control Register (MC_CGM_AC3_SC)

This register is used to select the current clock source for the PLL0 reference clock.

See Figure 27-8 for details.

Address: 0h base + 860h offset = 860h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							SELCTL	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC3_SC field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELCTL	Auxiliary Clock 3 Source Selection Control — This value selects the current source for auxiliary clock 3. 0 16 MHz IRCOSC 1 8-40 MHz XOSC
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.22 Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

Address: 0h base + 864h offset = 864h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							SELSTAT	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC3_SS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELSTAT	Auxiliary Clock 3 Source Selection Status — This value indicates the current source for auxiliary clock 3. 0 16 MHz IRCOSC 1 8-40 MHz XOSC
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.23 Auxiliary Clock 4 Select Control Register (MC_CGM_AC4_SC)

This register is used to select the current clock source for the PLL1 reference clock.

See Figure 27-9 for details.

Address: 0h base + 880h offset = 880h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						SELCTL		0							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC4_SC field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 SELCTL	Auxiliary Clock 4 Source Selection Control — This value selects the current source for auxiliary clock 4. 00 reserved 01 8-40 MHz XOSC 10 reserved 11 PLL0 PHI1
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.24 Auxiliary Clock 4 Select Status Register (MC_CGM_AC4_SS)

This register provides the current auxiliary clock 4 source selection.

Address: 0h base + 884h offset = 884h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						SELSTAT		0							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC4_SS field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 SELSTAT	Auxiliary Clock 4 Source Selection Status — This value indicates the current source for auxiliary clock 4. 00 reserved 01 8-40 MHz XOSC 10 reserved 11 PLL0 PHI1
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.25 Auxiliary Clock 5 Select Control Register (MC_CGM_AC5_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 5 divider 0: LFAST PLL clock

See Figure 27-10 for details.

Address: 0h base + 8A0h offset = 8A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	Auxiliary Clock 5 Source Selection Control — Selects the source for auxiliary clock 5. 000 reserved 001 8-40 MHz XOSC 010 PLL0 PHI 011 reserved 100 reserved 101 LFAST-SysClk pin 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.26 Auxiliary Clock 5 Select Status Register (MC_CGM_AC5_SS)

This register provides the current auxiliary clock 5 source selections.

Address: 0h base + 8A4h offset = 8A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	Auxiliary Clock 5 Source Selection Status — This value indicates the current source for auxiliary clock 5. 000 reserved 001 8-40 MHz XOSC 010 PLL0 PHI 011 reserved 100 reserved 101 LFAST-SysClk pin 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.27 Auxiliary Clock 5 Divider 0 Configuration Register (MC_CGM_AC5_DC0)

This register controls auxiliary clock 5 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8A8h offset = 8A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0										DIV				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 0 1 Enable auxiliary clock 5 divider 0
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DIV	Divider Division Value — The resultant LFAST PLL clock will have a period ’DIV + 1’ times that of auxiliary clock 5. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the LFAST PLL clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.28 Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 6 divider 0: CLKOUT pin clock

See Figure 27-11 for details.

Address: 0h base + 8C0h offset = 8C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC6_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	Auxiliary Clock 6 Source Selection Control — This value selects the current source for auxiliary clock 6. 000 16 MHz IRCOSC 001 8-40 MHz XOSC 010 PLL0 PHI 011 reserved 100 PLL1 PHI 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.29 Auxiliary Clock 6 Select Status Register (MC_CGM_AC6_SS)

This register provides the current auxiliary clock 6 source selection.

Address: 0h base + 8C4h offset = 8C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC6_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	Auxiliary Clock 6 Source Selection Status — This value indicates the current source for auxiliary clock 6. 000 16 MHz IRCOSC 001 8-40 MHz XOSC 010 PLL0 PHI 011 reserved

Table continues on the next page...

MC_CGM_AC6_SS field descriptions (continued)

Field	Description
	100 PLL1 PHI 101 reserved 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.30 Auxiliary Clock 6 Divider 0 Configuration Register (MC_CGM_AC6_DC0)

This register controls auxiliary clock 6 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8C8h offset = 8C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0								DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC6_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 6 divider 0 1 Enable auxiliary clock 6 divider 0
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	Divider Division Value — The resultant CLKOUT pin clock will have a period ’DIV + 1’ times that of auxiliary clock 6. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CLKOUT pin clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.31 Auxiliary Clock 10 Select Control Register (MC_CGM_AC10_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 10 divider 0: ENET_CLK

See Figure 27-12 for details.

Address: 0h base + 940h offset = 940h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELCTL	Auxiliary Clock 10 Source Selection Control — This value selects the current source for auxiliary clock 10. 000 16 MHz IRCOSC 001 8-40 MHz XOSC 010 PLL0 011 PLL0 PHI1 100 PLL1 101 RMII clock 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.32 Auxiliary Clock 10 Select Status Register (MC_CGM_AC10_SS)

This register provides the current auxiliary clock 10 source selection.

Address: 0h base + 944h offset = 944h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	Auxiliary Clock 10 Source Selection Status — This value indicates the current source for auxiliary clock 10. 000 16 MHz IRCOSC 001 8-40 MHz XOSC 010 PLL0 011 PLL0 PHI1 100 PLL1 101 RMII clock 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.33 Auxiliary Clock 10 Divider 0 Configuration Register (MC_CGM_AC10_DC0)

This register controls auxiliary clock 10 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 948h offset = 948h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 10 divider 0 1 Enable auxiliary clock 10 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider Division Value — The resultant motor control clock will have a period 'DIV + 1' times that of auxiliary clock 10. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the motor control clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.34 Auxiliary Clock 11 Select Control Register (MC_CGM_AC11_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 11 divider 0: ENET_TIME_CLK

See Figure 27-13 for details.

Address: 0h base + 960h offset = 960h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELCTL			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC11_SC field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_CGM_AC11_SC field descriptions (continued)

Field	Description
5–7 SELCTL	Auxiliary Clock 11 Source Selection Control — This value selects the current source for auxiliary clock 11. 000 16 MHz IRCOSC 001 8-40 MHz XOSC 010 PLL0 011 PLL0 PHI1 100 PLL1 101 RMII clock 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.35 Auxiliary Clock 11 Select Status Register (MC_CGM_AC11_SS)

This register provides the current auxiliary clock 11 source selection.

Address: 0h base + 964h offset = 964h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SELSTAT			0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC11_SS field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SELSTAT	Auxiliary Clock 11 Source Selection Status — This value indicates the current source for auxiliary clock 11. 000 16 MHz IRCOSC 001 8-40 MHz XOSC 010 PLL0 011 PLL0 PHI1 100 PLL1 101 RMII clock 110 reserved 111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.36 Auxiliary Clock 11 Divider 0 Configuration Register (MC_CGM_AC11_DC0)

This register controls auxiliary clock 11 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 968h offset = 968h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC11_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 11 divider 0 1 Enable auxiliary clock 11 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider Division Value — The resultant motor control clock will have a period ’DIV + 1’ times that of auxiliary clock 11. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the motor control clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.4 Functional Description

27.4.1 System Clock Generation

The following figure shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the 16 MHz IRCOSC as the system clock and to ignore the system clock select.

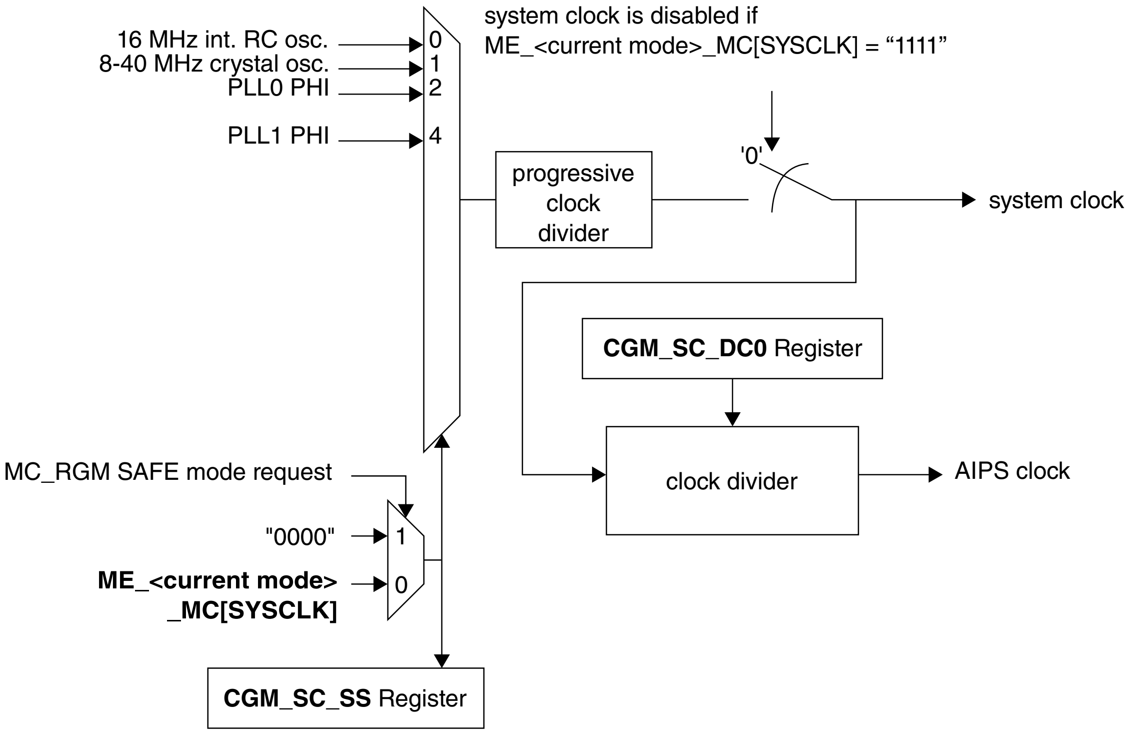


Figure 27-2. MC_CGM System Clock Generation Overview

27.4.1.1 System Clock Source Selection

During normal operation, the system clock selection is controlled

- on a SAFE mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

27.4.1.2 Progressive System Clock Switching

Based on the power level values of the current and target modes, the MC_ME requests the MC_CGM to ramp the system clock frequency down and/or up. During ramp-down, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVEn registers, where n corresponds to the current system clock source selection. During ramp-up, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVSn registers, where *n* corresponds to the target system clock source selection.

27.4.1.2.1 Generic Clock Change Requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in the following equation.

$$a_{\mathrm{max}} = \frac{f_{\mathrm{chg}}}{f_{\mathrm{src}}}$$

Equation 10

27.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field CGM_PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of the 16 MHz internal RC oscillator. After the expiration of this time, the module changes the clock divider value, which changes the frequency of the system clock.

27.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the PCS_DIVCn[RATE] register is given in the following table.

Table 27-1. MC_CGM CGM_PCS_DIVC*n*[RATE] values

a_{max}	d	CGM_PCS_DIVC*n*[RATE]
0.05	0.012	12
0.10	0.048	48
0.15	0.112	112
0.20	0.184	184

27.4.1.2.4 Generic Clock Change Properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in the following equation.

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16\text{MHz}}\right)}{d}} \right\rceil$$

Equation 11

27.4.1.2.5 Clock Ramp-Down

The clock ramp-down starts with the divider value 1 and with the given divider increment value PCD_DIVCn[RATE] and ends with the given divider value PCS_DIVEn[DIVE].

The divider end value for clock ramp-down is given in the following equation.

$$\text{PCS\_DIVEn}[\text{DIVE}] = \frac{f_{src}}{16\text{MHz}} \times 1000 - 1$$

Equation 12

Where f_{curr} is the frequency of the currently selected system clock source.

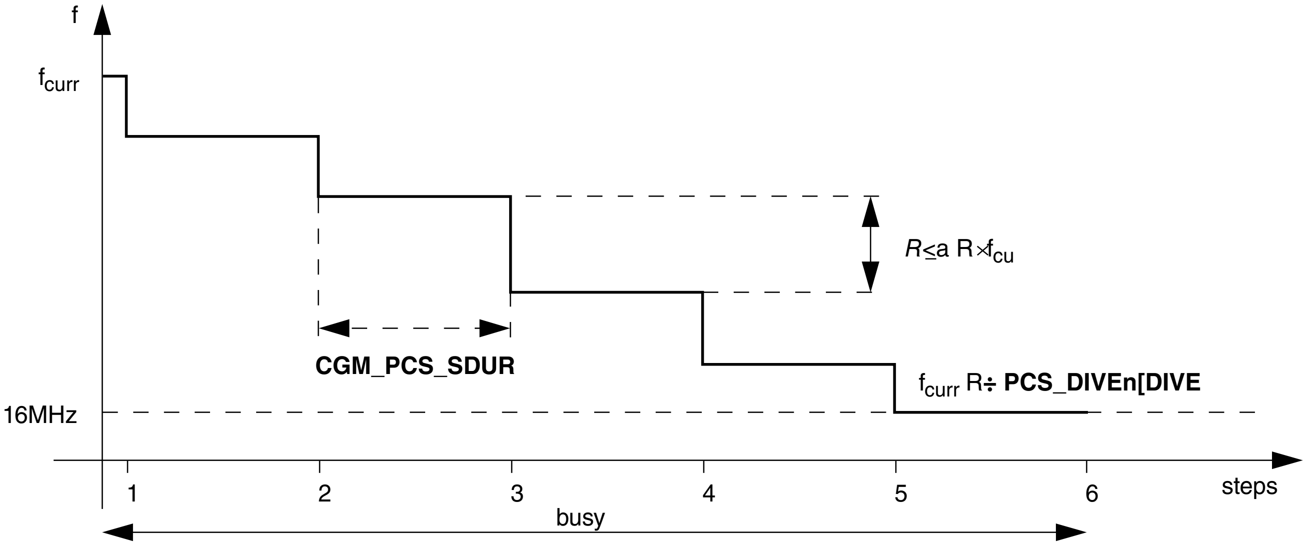


Figure 27-3. MC_CGM System Clock Ramp-Down Timing (k = 6 example)

27.4.1.2.6 Clock Ramp-Up

The clock ramp-up starts with the given divider value PCS_DIVSn[DIVS] and with the given divider decrement value PCD_DIVCn[INIT] and ends with the divider value 1.

The initial divider change start value PCS_DIVCn[INIT] for system clock ramp-up is given in Equation 13 on page 829.

$$\text{PCS\_DIVC}n[\text{INIT}] = d \times k \times 1000$$

Equation 13

Where k is calculated from Equation 11 on page 828 using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in Equation 14 on page 829.

$$\text{PCS\_DIVS}n[\text{DIVS}] = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

Equation 14

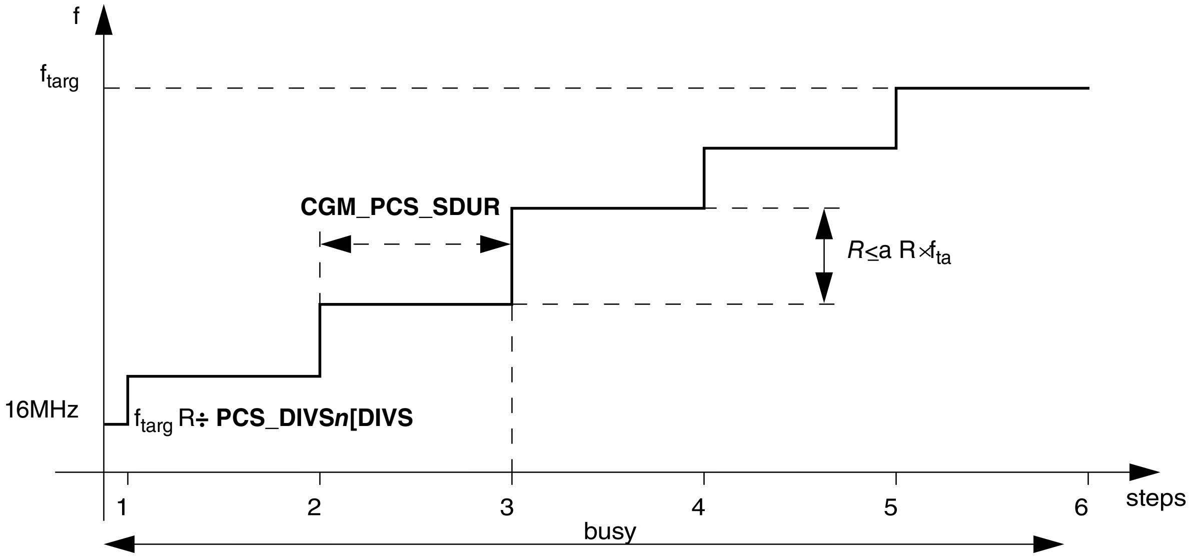


Figure 27-4. MC_CGM System Clock Ramp-Up Timing (k = 6 example)

27.4.1.3 System Clock Disable

During the TEST mode, the system clock can be disabled by the MC_ME.

27.4.1.4 System Clock Dividers

The MC_CGM generates the following derived clock from the system clock:

- AIPS clock - controlled by the CGM_SC_DC0 register

27.4.2 Auxiliary Clock Generation

Figure 27-5 through Figure 27-11 show the block diagrams of the generation logic for the various auxiliary clocks. See the following sections for auxiliary clock selection control:

- Auxiliary Clock 0 Select Control Register (MC_CGM_AC0_SC)
- Auxiliary Clock 3 Select Control Register (MC_CGM_AC3_SC)
- Auxiliary Clock 4 Select Control Register (MC_CGM_AC4_SC)
- Auxiliary Clock 5 Select Control Register (MC_CGM_AC5_SC)

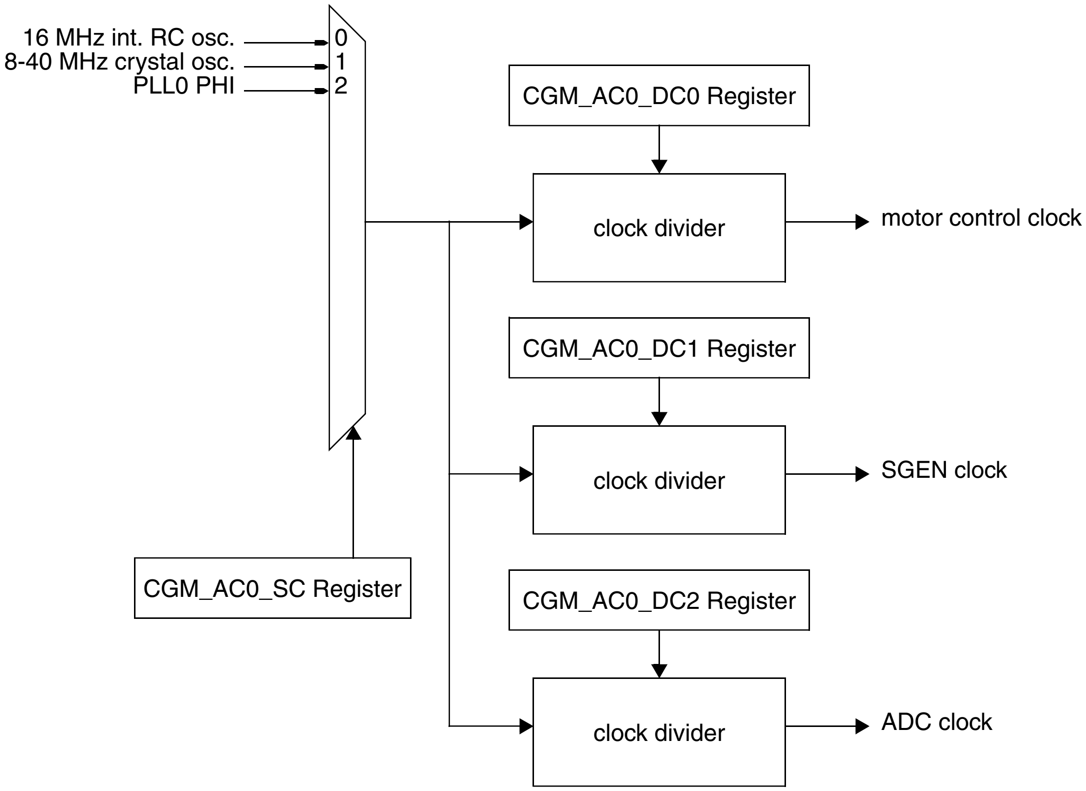


Figure 27-5. MC_CGM Auxiliary Clock 0 Generation Overview

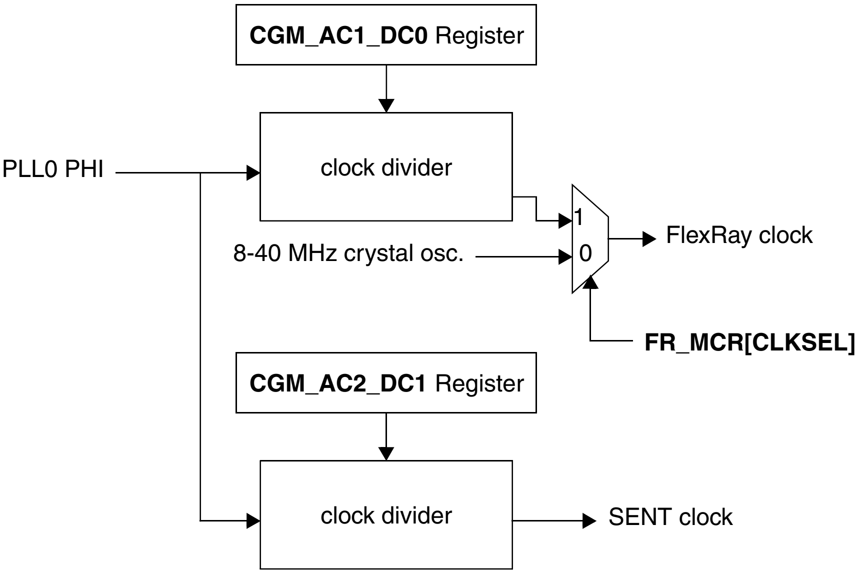


Figure 27-6. MC_CGM Auxiliary Clock 1 Generation Overview

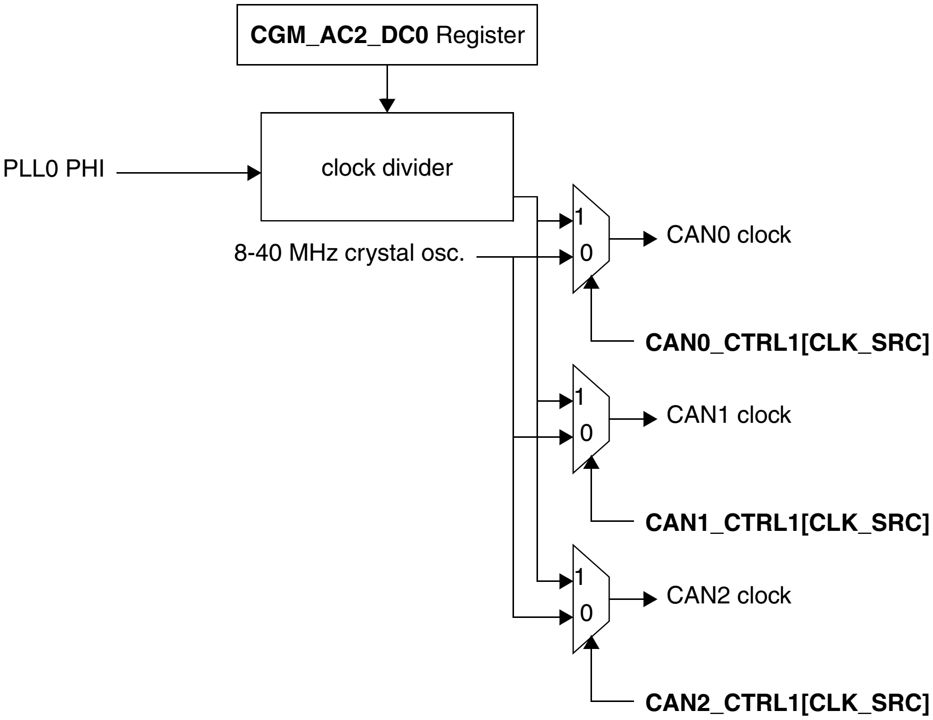


Figure 27-7. MC_CGM Auxiliary Clock 2 Generation Overview

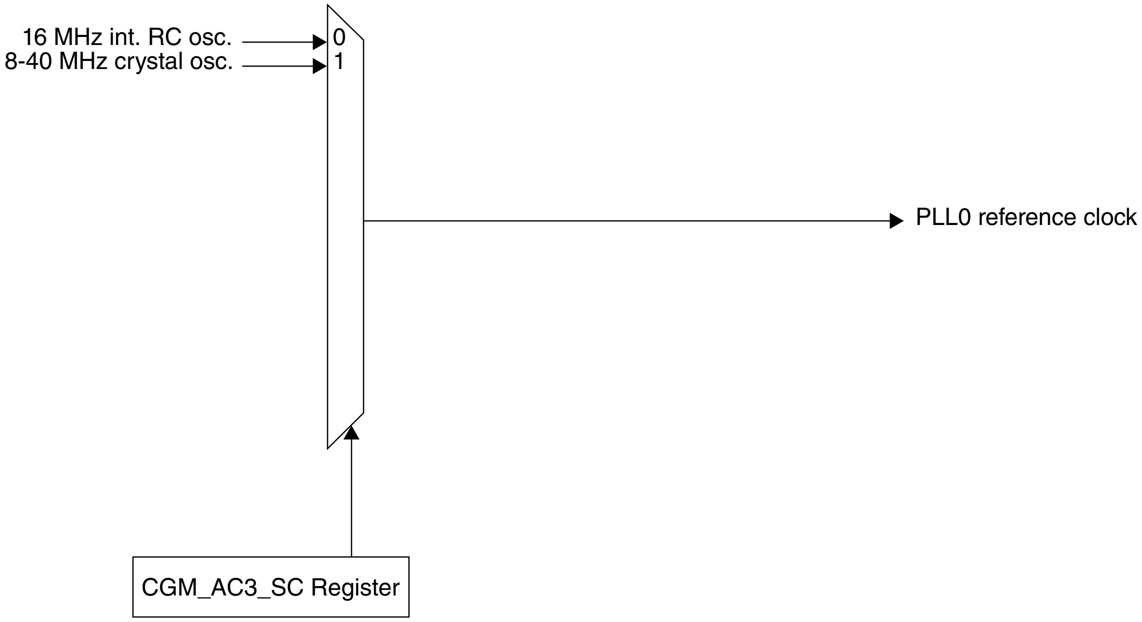


Figure 27-8. MC_CGM Auxiliary Clock 3 Generation Overview

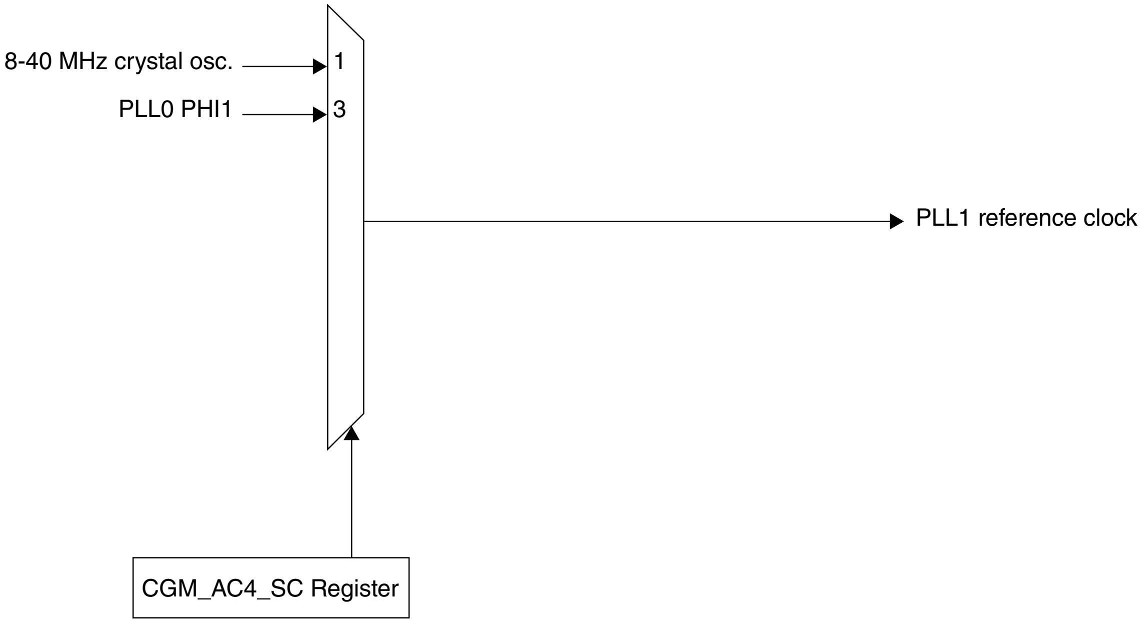


Figure 27-9. MC_CGM Auxiliary Clock 4 Generation Overview

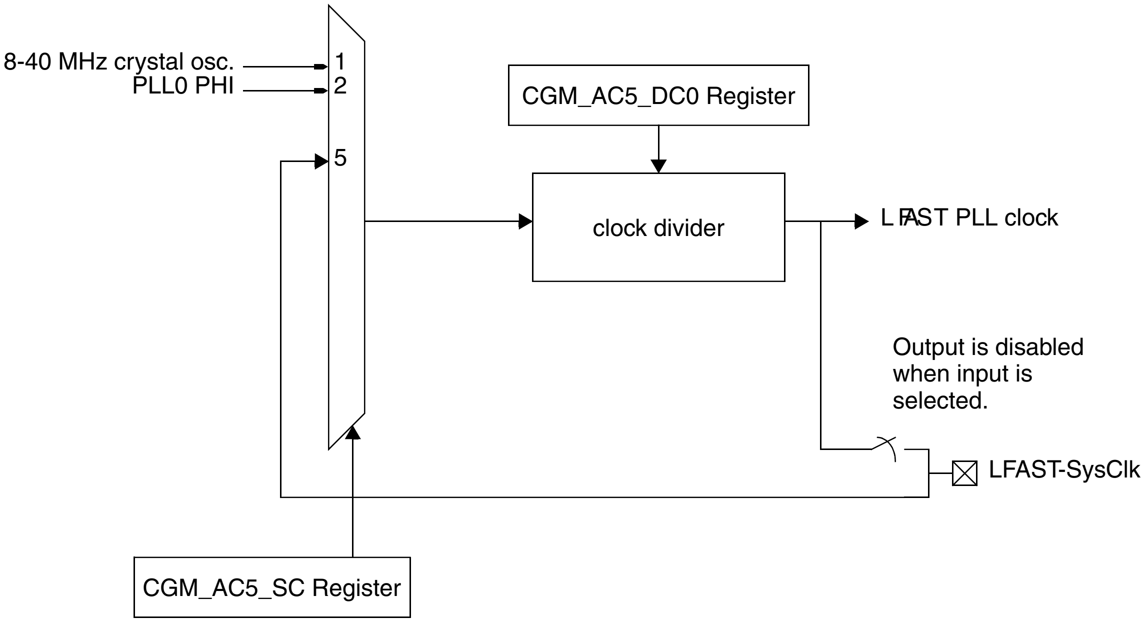


Figure 27-10. MC_CGM Auxiliary Clock 5 Generation Overview

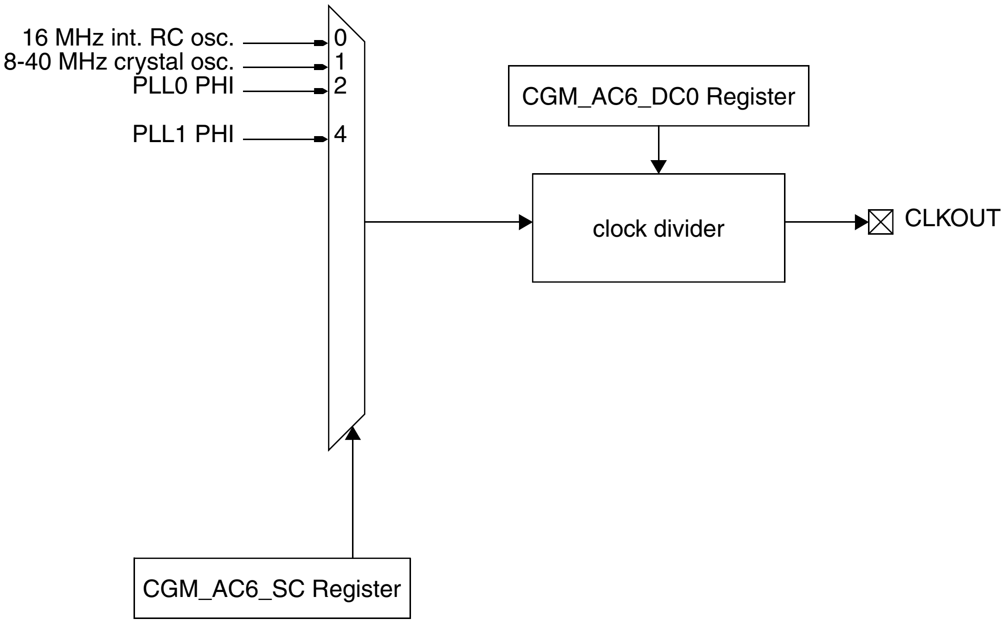


Figure 27-11. MC_CGM Auxiliary Clock 6 Generation Overview

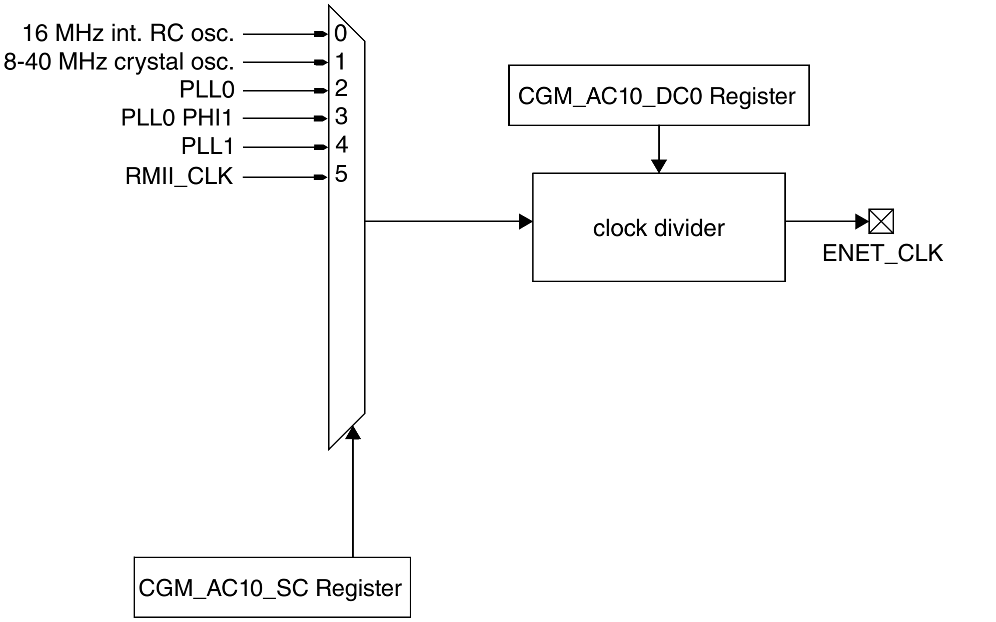


Figure 27-12. MC_CGM Auxiliary Clock 10 Generation Overview

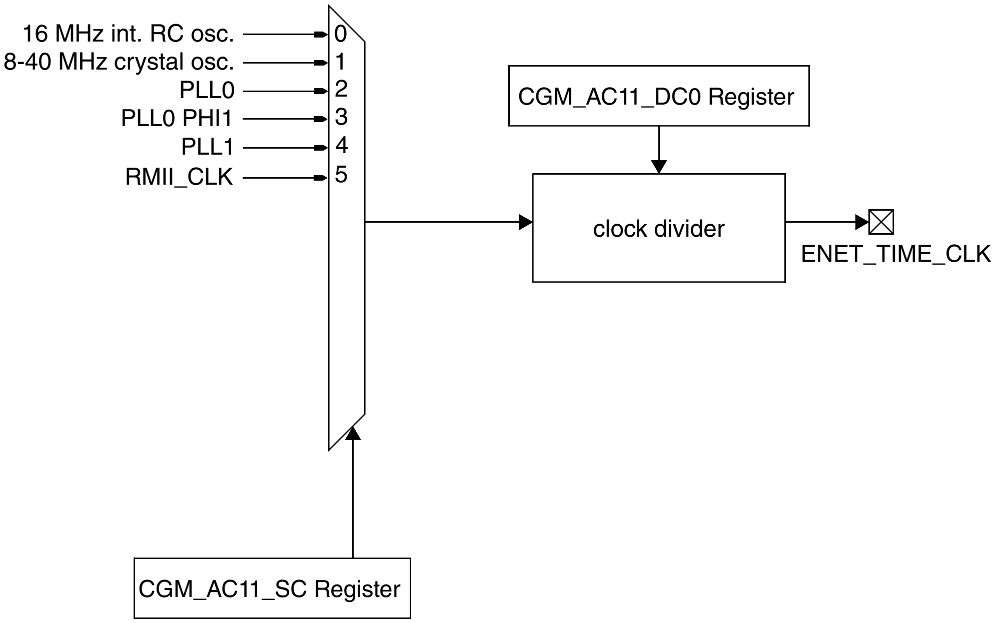


Figure 27-13. MC_CGM Auxiliary Clock 11 Generation Overview

27.4.2.1 Auxiliary Clock Dividers

The MC_CGM generates the following derived clocks:

- Motor Control clock - controlled by the **CGM_AC0_DC0** register
- SGEN clock - controlled by the **CGM_AC0_DC1** register
- ADC clock - controlled by the **CGM_AC0_DC2** register
- FlexRay clock - controlled by the CGM_AC1_DC0 register
- SENT clock - controlled by the CGM_AC1_DC1 register
- CAN clock - controlled by the CGM_AC2_DC0 register
- LFAST PLL clock - controlled by the CGM_AC5_DC0 register
- CLKOUT pin clock - controlled by the CGM_AC6_DC0 register
- ENET clock - controlled by the CGM_AC10_DC0 register
- ENET TIME clock - controlled by the CGM_AC11_DC0 clock register

27.4.3 Dividers Functional Description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)
- Auxiliary Clock 0 Divider 0 Configuration Register (MC_CGM_AC0_DC0)
- Auxiliary Clock 0 Divider 1 Configuration Register (MC_CGM_AC0_DC1)
- Auxiliary Clock 0 Divider 2 Configuration Register (MC_CGM_AC0_DC2)
- Auxiliary Clock 1 Divider 0 Configuration Register (MC_CGM_AC1_DC0)
- Auxiliary Clock 1 Divider 1 Configuration Register (MC_CGM_AC1_DC1)
- Auxiliary Clock 2 Divider 0 Configuration Register (MC_CGM_AC2_DC0)
- Auxiliary Clock 5 Divider 0 Configuration Register (MC_CGM_AC5_DC0)
- Auxiliary Clock 6 Divider 0 Configuration Register (MC_CGM_AC6_DC0)
- Auxiliary Clock 10 Divider 0 Configuration Register (MC_CGM_AC10_DC0)
- Auxiliary Clock 11 Divider 0 Configuration Register (MC_CGM_AC11_DC0)

The reset value of all counters is '1'. If a divider has its DE bit in the respective configuration register set to '0' (the divider is disabled), any value in its DIV field is ignored.

Chapter 28
OSC Digital Interface (XOSC)

28.1 Introduction

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provide the register interface for the programmable features. Selection of the XOSC as a source clock is controlled by the Clock Generation Module (MC_CGM). The XOSC is enabled and disabled by configuring the Mode Entry Module (MC_ME).

The programmable features of the XOSC digital interface are as follows:

- Oscillator power-down control and status
- Oscillator startup delay
- Oscillator clock available interrupt
- Oscillator bypass mode control
- Crystal oscillator mode selection
  - Loop Controlled Pierce Mode (LCP)
  - Full Swing Pierce Mode (FSP)

28.2 Functional description

The XOSC digital interface provides control of the on-chip oscillator and the register interface for the mode and startup delay control. The oscillator provides an output clock that is used as an input to the PLLs and can be selected as the reference clock for many of the peripherals on the device.

The table below shows the possible configurations of the XOSC.

Table 28-1. XOSC configurations

OSC EN (from MC_ME)	XOSC_CTL[OSCBYP]	XTAL	EXTAL	XOSC Output	XOSC MODE
0	0	Crystal, or left opened	Crystal, or left opened	0	Power down, IDDQ
0	1	x	external clock	Undefined	Reserved, OSC Disabled
1	1	x	external clock	XTALOUT	Bypass, OSC Enabled
1	0	crystal	crystal	XTALOUT	Normal, OSC Enabled

28.2.1 Oscillator power-down control and status

Enabling and disabling of the oscillator is done in the Mode Entry (MC_ME) module (see “Mode Entry Module (MC_ME)” chapter). The ME_<*mode*>_MC[XOSCON] controls the power down of oscillator based on the current device mode (see “MC_ME Memory Map” table for available *modes*). The status of the XOSC is shown in the MC_ME_GS[S_XOSC].

28.2.2 Oscillator startup delay

A bit in the UTEST flash memory determines whether the analog portion of the oscillator is enabled at powerup. If enabled, the analog portion of the oscillator is powered-up during Phase3 of the device reset sequence (pre-self-test sequence). In order to allow the oscillator to reach full amplitude before use, the internal counter (OSCCNT) is started when the device exits reset. The counter is driven by the oscillator output clock. When the counter reaches XOSC_CTL[EOCV] × 512, a signal is sent to the MC_ME module, allowing a mode switch to the oscillator (when used as a system clock or an input to PLL0).

The default value for this field after reset depends on implementation and can be found in the XOSC Device Configuration. After reset, software can clear or change the EOCV field in order to shorten or lengthen the delay until the oscillator is ready.

Note

It is advised that if the EOCV value is modified it should be done as early as possible in user code. This will ensure that the new value will be used before OSCCNT reaches the count value.

28.2.3 Oscillator clock available interrupt

An interrupt can be generated when the count value is reached (OSCCNT = XOSC_CTL[EOCV] × 512). The XOSC_CTL[I_OSC] will be set and an interrupt will be generated if the interrupt mask is set (XOSC_CTL[M_OSC] = 1).

28.2.4 Oscillator bypass mode

The oscillator circuit can be bypassed by setting XOSC_CTL[OSCBYP]. This field can only be set by software and is only cleared by a system reset. In bypass mode, the oscillator is disabled and the input clock on the EXTAL pins is level-shifted and driven to the logic on the device. The bypass configuration is independent of the power-down mode of the oscillator.

NOTE

The external oscillator must be running before the XOSC is turned off. If XOSC is being turned on, the status will be updated only after the clock starts toggling.

NOTE

Power down from the MC_ME will have no effect on the oscillator. The bypass clock will continue running in the device. It is ensured that mode transitions from the MC_ME complete successfully by providing bypass information to the MC_ME.

28.2.5 Crystal oscillator mode selection

There are two oscillator mode selections possible, Full Swing Pierce mode and Loop Controlled Pierce mode. The operating mode is selected by configuring the XOSC_CTL[OSCM].

28.2.5.1 Loop Controlled Pierce mode

Loop Controlled Pierce mode (LCP) is the default oscillator mode. LCP has the following attributes:

- Provides a smaller frequency range

- Limits the amplitude of the crystal oscillation utilizing Amplitude Loop Control (ALC)
- Reduced EMI
- Reduced power consumption
- Shorter settling time
- Crystal is exposed to reduced stress since the amplitude is reduced
- Reduced number of external components, feedback resistor already integrated

28.2.5.2 Full Swing Pierce mode

Full Swing Pierce mode (FSP) has the following attributes when compared with LCP:

- Provides a wider frequency range
- Less susceptible to noise
- The crystal amplitude is rail-to-rail
  - Distorted oscilation with several harmonics
  - Increased EMI
- Slower settling time
- Feedback resistor and 300 ohms of current limitation are already integrated.

NOTE

> Consult with crystal supplier to determine the need of extra resistance in XTAL to avoid crystal overdrive.

28.3 Memory map and register definition

XOSC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XOSC Control Register (XOSC_CTL)	32	R/W	See section	28.3.1/841

28.3.1 XOSC Control Register (XOSC_CTL)

NOTE

The XOSC_CTL is only writable in supervisor mode.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OSCBYP	OSCM	MON	Reserved					EOCV							
W																
Reset	*	*	*	0	0	0	0	0	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC	0		Reserved					I_OSC	0					0	0
W									w1c							
Reset	*	0	0	0*	0*	0*	0*	0*	0	0	0	0	0	0	0	0

* Notes:

- Reserved field: This field can be written at any time, but writes have no effect. Reads return the previously written value.
- M_OSC field: See Clocking chapter for reset value.
- EOCV field: See Clocking chapter for reset value.
- MON field: See Clocking chapter for reset value.
- OSCM field: See Clocking chapter for reset value.
- OSCBYP field: See Clocking chapter for reset value.

XOSC_CTL field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL is used as root clock.

Table continues on the next page...

XOSC_CTL field descriptions (continued)

Field	Description
1 OSCM	Crystal Oscillator Mode. This bit selects the oscillator mode when not in bypass. 0 FSP mode (full swing pierce mode) 1 LCP mode (loop controlled pierce mode)
2 MON	XOSC clock monitor enable. 0 Monitoring is disabled. 1 Monitoring is enabled.
3–7 Reserved	This field is reserved.
8–15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV × 512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 Reserved	This field is reserved. This field can be written at any time, but writes have no effect. Reads return the previously written value.
24 I_OSC	Crystal oscillator clock interrupt This bit is set by hardware when OSCCNT counter reaches the count value EOCV x 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.
25–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Chapter 29
IRCOSC Digital Interface

29.1 Introduction

The Internal RC Oscillator Digital Interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC), as well as an internal temperature sensor and an internal voltage regulator used to compensate external variations in temperature and voltage. It also provides access to trimming information used for safety implementations.

29.2 Functional description

The IRCOSC provides a high frequency (f_{MRC}) clock with a nominal frequency of 16 MHz. After deassertion of reset, the output clock is available after a short time (see the Data Sheet for timing details) which is required for stabilization. The IRCOSC status is updated in the MC_ME_GS[S_IRC] field (see the "Mode Entry Module MC_ME" chapter).

The IRCOSC output frequency can be trimmed by configuring IRCOSC_CTL[USER_TRIM[4:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with δf_{var_SW}, as defined in the Data Sheet.

29.3 Memory map and register definition

IRCOSC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	IRCOSC Control Register (IRCOSC_CTL)	32	R/W	See section	29.3.1/844

29.3.1 IRCOSC Control Register (IRCOSC_CTL)

The IRCOSC_CTL contains user programmable parameters.

IRCOSC_CTL is writable only in supervisor mode.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							Reserved	0			USER_TRIM				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			Reserved					0		Reserved	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IRCOSC_CTL field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This bit can be written with any value, but writes are ignored. A read returns last written value.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 USER_TRIM	User trimming bits with respect to nominal factory frequency The MSB of the USER_TRIM bits is used to determine whether the frequency will be increased or decreased. If MSB = 0 then a change in USER_TRIM[3:0] will decrease the frequency, and likewise, if MSB = 1 then a change in USER_TRIM[3:0] will increase the frequency. Frequency trimming calculation shows how RCOSC_CTL[USER_TRIM] field is used to trim the IRCOSC frequency.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

IRCOSC_CTL field descriptions (continued)

Field	Description
19–23 Reserved	This field is reserved. This field can be written at any time, but writes have no effect. Reads return the previously written value.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This field may sometimes have the value of 1, and can be cleared by writing a 1 to this bit. However, this field has no effect on the IRCOSC or chip behavior.
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.3.2 Frequency trimming calculation

Table 29-1. IRCOSC_CTL[USER_TRIM] frequency trimming calculation

USER_TRIM[4:0] value	Frequency
....	....
10011	$f + (3 \times \delta f_{TRIM})$
10010	$f + (2 \times \delta f_{TRIM})$
10001	$f + \delta f_{TRIM}$
10000	f
00000	f
00001	$f - \delta f_{TRIM}$
00010	$f - (2 \times \delta f_{TRIM})$
00011	$f - (3 \times \delta f_{TRIM})$
....	....

NOTE

See the data sheet for δf_{TRIM} value and min/max frequency variations.

Chapter 30
RAM Controller (PRAMC)

30.1 Introduction

This section provides an overview of the Platform RAM Controller. The RAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The RAM controller supports two 64-bit AHB interfaces and a 64-bit RAM array interface. The primary AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The backdoor AHB port provides a connection from the flash controller to facilitate calibration overlay access. Shown below in Figure 30-1 is a simplified block diagram depicting the position of the RAM controller within a typical platform architecture.

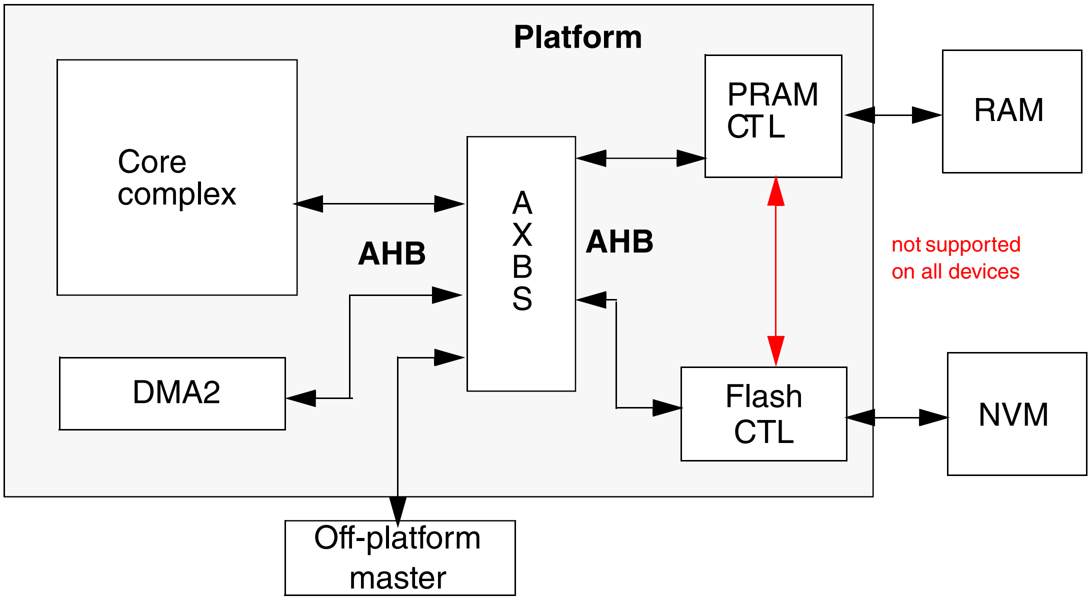


Figure 30-1. Simplified platform block diagram

The following list summarizes the key features of the RAM controller:

- System bus supports 64-bit data + 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data + 8-bit ECC interface
- Late-write support via 64-bit data + 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than a doubleword

The address path of the RAM controller contains a mux that chooses among the addresses presented on the system bus for a read request, either the address from the temporary holding register or the address from the late-write buffer. The temporary holding register contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to present simultaneously the address and associated write data, which is not valid until the AHB dataphase. The late-write buffer holds write requests which are delayed to facilitate single-cycle response on the system bus.

The read datapath contains a mux that chooses the source of the read data to be presented on the system bus on a read request. In the event that a read request matches the contents of the late-write buffer, a RAM access is not required and the late-write buffer contents are selected onto the read databus. Otherwise, the read data is a result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the RAM controller, including the late-write storage buffer, to support end-to-end ECC (e2eECC) coverage.

The write datapath contains the mux that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and instead be stored directly, precisely in the RAM.

30.2 SRAM controller memory map and register definition

The RAM controller module provides an IPS programming model mapped to a standard on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses generate an IPS error termination.

NOTE

Attempted updates to the PRAMC programming model while a PRAMC operation is in progress will result in non-deterministic behavior. Software must be architected to avoid this scenario by ensuring that PRAMC configuration changes are made only during system boot or when only one master is enabled. In multi-core devices, update the PRAMC configuration only when one core is active and no other masters, e.g., DMA or communications modules, are enabled. Move any instruction execution or memory references outside the system RAM while updating the PRAMC configuration, e.g., to the core local memory space.

PRAMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	32	R/W	0000_0200h	30.2.1/849

30.2.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)

The Platform RAM Configuration Register 1 (PRCR1) is used to specify operation of the RAM controller.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						PRI		P1_ BO_ DIS	P0_ BO_ DIS	0					FT_ DIS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

PRAMC_PRCR1 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PRAMC_PRCR1 field descriptions (continued)

Field	Description
22–23 PRI	AHB port arbitration mode. Use this register to select the AHB port arbitration mode. 00 Round Robin arbitration is selected. 01 Port p0 has priority over port p1. 10 Port p1 has priority over port p0. 11 Reserved.
24 P1_BO_DIS	Port p1 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

30.3 Functional description

This section describes the functions of the RAM controller.

30.3.1 Read and Write operations

The RAM controller processes read and write requests to on-chip RAM and provides a register interface for access to performance tuning functions.

NOTE

The RAM controller register interface is accessible only in supervisor mode. Accesses in user mode return a transfer error.

30.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the system bus.

30.3.1.2 Optional read wait-state

The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCRx[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCRx[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCRx[FT_DIS]=0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if PRCRx[FT_DIS]=1, and a WRAP4 burst will have an access time of 3-2-2-2.

NOTE

The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency.

30.3.2 Writes

This section discusses various write operations of the RAM controller.

30.3.2.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back to back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Rather, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

30.3.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with the potentially corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. Figure 30-2 provides details on the read-modify-write datapath. For details on the ECC coding scheme, refer to Reliability considerations.

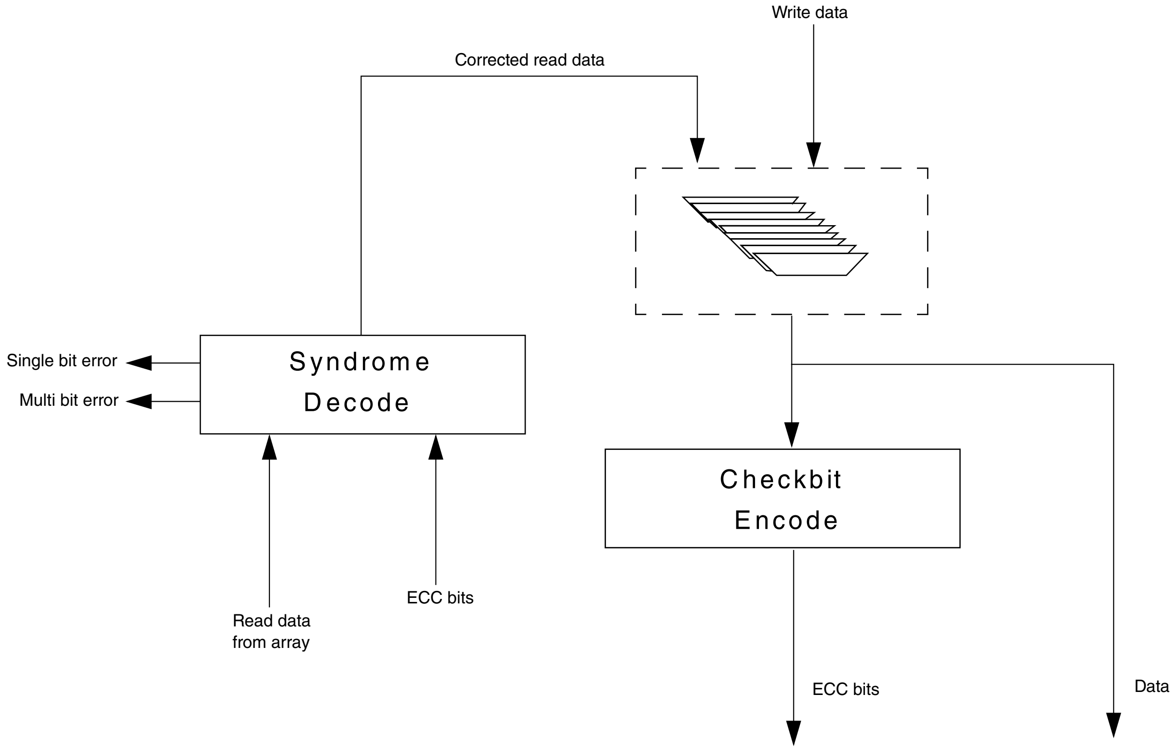


Figure 30-2. Read-modify-write datapath

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of size less than 64 bits require the insertion of one wait state before the data phase can be completed.

30.3.2.3 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

NOTE

Unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

30.4 Initialization/application information

It is essential that each memory address be written to a known value before it is read, to initialize the ECC. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Without writing an address to a known value first, a read from this address will most likely generate an uncorrectable ECC event.

One possibility for initializing PRAMC memory space is to use a stored 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

30.5 Reliability considerations

This section discusses reliability considerations of the RAM controller.

30.5.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named for M.Y. Hsiao, an IBM researcher who published extensively in the early 1970s on SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns.
2. Every column is distinct.
3. Every column contains an odd number of ones, and hence is "odd weight".

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field.

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in the following table. A '*' in the table indicates the corresponding data or address bit is XOR'ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR'ed) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit.*

Table 30-1. RAM controller basic H-matrix definition

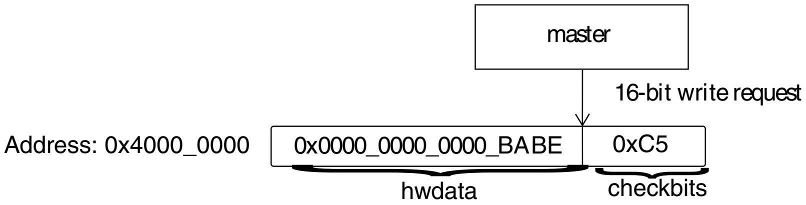
		XOR sum	Data Bit																															
			Byte 7								Byte 6								Byte 5								Byte 4							
			63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Checkbits [7:0]	7		*				*					*			*		*	*		*	*	*	*					*	*			*	*	
	6				*	*			*	*					*		*		*				*				*			*	*			*
	5		*	*	*	*					*				*	*		*		*	*				*	*	*				*			
	4		*		*			*		*					*		*		*			*			*	*			*				*	
	3										*	*	*	*	*		*				*		*				*	*	*	*				
	2						*		*					*		*			*	*				*	*							*		*
	1			*		*		*		*	*	*	*	*						*		*		*		*						*		*
	0			*			*	*	*				*			*	*	*		*				*				*		*	*		*	
		XOR sum	Byte 3								Byte 2								Byte 1								Byte 0							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7						*		*				*		*				*	*			*	*			*				*	*		*
	6		*	*		*	*				*				*				*	*		*	*					*	*				*	*
	5				*	*			*	*	*			*	*	*					*		*	*		*								
	4		*		*	*				*		*	*				*	*	*			*			*	*				*		*		
	3		*	*	*			*						*	*		*	*			*				*		*	*	*			*		
	2			*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
	1			*						*		*			*	*	*			*				*				*			*	*	*	
	0			*		*		*		*				*				*				*				*				*	*		*	*
		XOR sum	Address Bit																															
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7			*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*		Unused	Unused	Unused
	6			*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
	5			*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*			
	4		*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*				
	3		*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*			
	2		*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*			
	1		*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*			
	0		*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*			

30.5.1.1 e2eECC considerations on less-than-64-bit write transactions

Recall writes of size less than 64-bit incur a read-modify-write action as a consequence of the ECC coding scheme 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The RAM controller uses the following checkbit manipulation technique to ensure any faults in the PRAMC control logic associated with performing the read-modify-write action will ultimately be detectable by e2eECC:

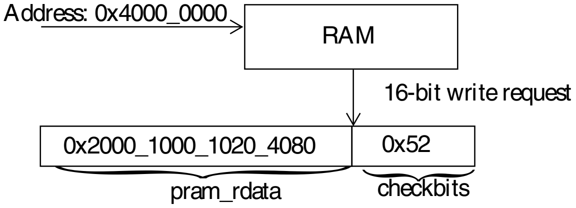
- On an 8-, 16-, or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non-pertinent byte lanes zeroed out, and the associated checkbit is based on this "zero-padded" write data.

  Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

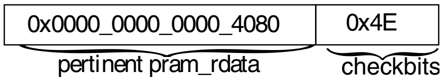


- The RAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

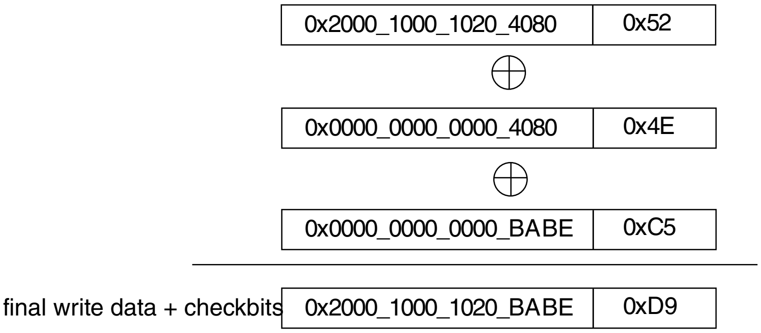
  If a non-correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.



- The RAM controller isolates the pertinent byte lanes of the read data by zero-padding the non-pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.



- Merge the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and RAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The RAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

30.5.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the RAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions:

- Direct system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

As a countermeasure against false misses to the late-write buffer on reads, the transaction monitor also verifies the expected source of data returned on a read as supplied either from the RAM or from the late-write buffer.

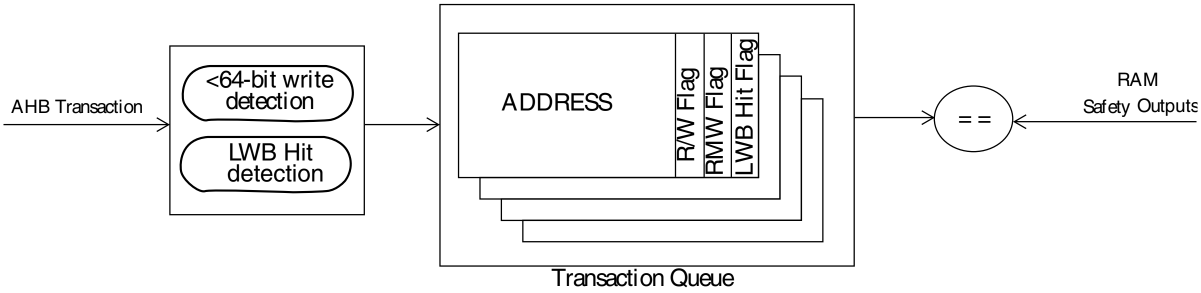


Figure 30-3. RAM controller transaction monitor block diagram

Chapter 31
Flash Memory Controller (PFLASH)

31.1 Introduction

The flash memory controller has these functions:

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash memory array.
- It serves as the interface to on-chip System RAM that can be used as overlay RAM.

The flash memory controller supports one 64-bit AHB bus and a 256-bit read data interface from each flash memory array. The AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to the AHB upon completion. The mini-cache entry is 256 bits in size, matching the flash array page size and providing 256 bytes of total buffered storage. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage.

31.2 Features

The following list summarizes the key features of the flash memory controller:

- One 64-bit AHB interface port (p0) allowing access to dedicated prefetch mini-cache.
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle "buffer hit" read response.

- Configurable access control based on read/write and AHB master ID attributes.
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets.
- Optional address pipelining capability to maximize throughput.
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary.
- Configurable overlay remapping of logical flash accesses to on-chip system RAM

31.3 Block diagrams

Figure 31-1 illustrates the connections of the flash memory controller and flash memory array within the chip.

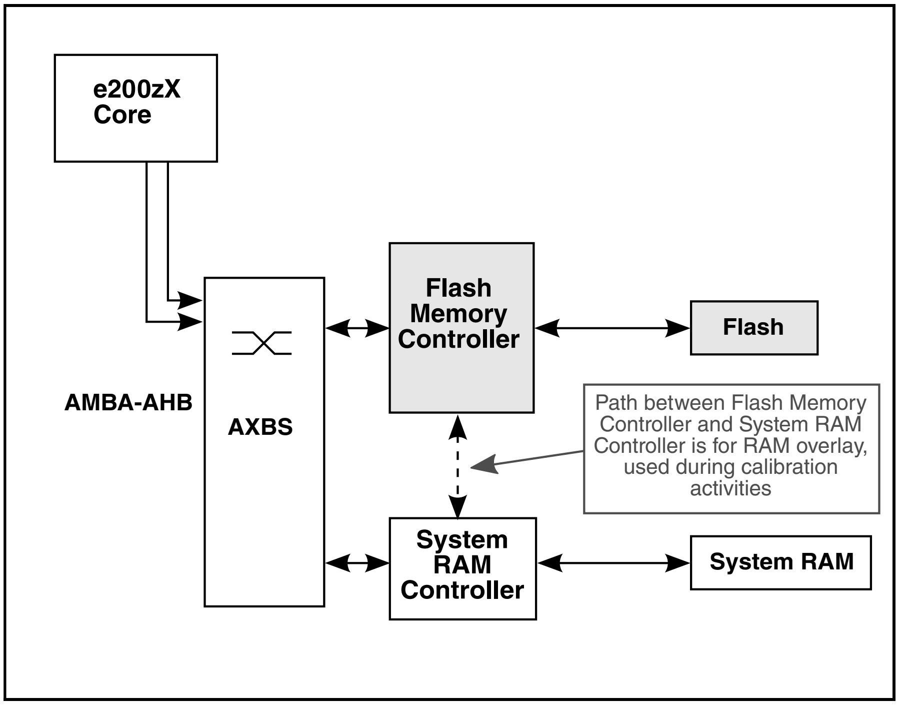

e200zX Core
AMBA-AHB
AXBS
Flash Memory Controller
Flash
Path between Flash Memory Controller and System RAM Controller is for RAM overlay, used during calibration activities
System RAM Controller
System RAM

Figure 31-1. Platform-centric simple block diagram with flash memory controller

31.4 Flash memory controller memory map

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash memory access configurationand overlay remapping configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the flash memory controller module is in the midst of an operation will result in non-deterministic behavior. Software must be architected to avoid this scenario. There is no idle indicator for the flash controller. The recommended flow for multi-core devices is to start only one core and execute initialization code to completion before starting the remaining cores. If the user needs to reconfigure the flash, code execution must be temporarily moved to system RAM.

PFLASH memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform Flash Configuration Register 1 (PFLASH_PFCR1)	32	R/W	0000_0601h	31.4.1/862
8	Platform Flash Configuration Register 3 (PFLASH_PFCR3)	32	R/W	0000_0000h	31.4.2/866
C	Platform Flash Access Protection Register (PFLASH_PFAPR)	32	R/W	FFFF_FFFFh	31.4.3/867
10	Platform Flash Remap Control Register (PFLASH_PFCRCR)	32	R/W	0000_0000h	31.4.4/870
14	Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)	32	R/W	0000_0000h	31.4.5/872
100	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD0_Word0)	32	R/W	0000_0000h	31.4.6/874
104	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD0_Word1)	32	R/W	0000_0000h	31.4.7/874
108	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD0_Word2)	32	R/W	0000_0000h	31.4.8/876
110	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD1_Word0)	32	R/W	0000_0000h	31.4.6/874
114	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD1_Word1)	32	R/W	0000_0000h	31.4.7/874
118	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD1_Word2)	32	R/W	0000_0000h	31.4.8/876
120	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD2_Word0)	32	R/W	0000_0000h	31.4.6/874
124	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD2_Word1)	32	R/W	0000_0000h	31.4.7/874
128	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD2_Word2)	32	R/W	0000_0000h	31.4.8/876
130	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD3_Word0)	32	R/W	0000_0000h	31.4.6/874
134	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD3_Word1)	32	R/W	0000_0000h	31.4.7/874

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
138	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD3_Word2)	32	R/W	0000_0000h	31.4.8/876
140	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD4_Word0)	32	R/W	0000_0000h	31.4.6/874
144	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD4_Word1)	32	R/W	0000_0000h	31.4.7/874
148	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD4_Word2)	32	R/W	0000_0000h	31.4.8/876
150	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD5_Word0)	32	R/W	0000_0000h	31.4.6/874
154	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD5_Word1)	32	R/W	0000_0000h	31.4.7/874
158	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD5_Word2)	32	R/W	0000_0000h	31.4.8/876
160	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD6_Word0)	32	R/W	0000_0000h	31.4.6/874
164	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD6_Word1)	32	R/W	0000_0000h	31.4.7/874
168	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD6_Word2)	32	R/W	0000_0000h	31.4.8/876
170	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD7_Word0)	32	R/W	0000_0000h	31.4.6/874
174	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD7_Word1)	32	R/W	0000_0000h	31.4.7/874
178	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD7_Word2)	32	R/W	0000_0000h	31.4.8/876

31.4.1 Platform Flash Configuration Register 1 (PFLASH_PFCR1)

The PFlash Configuration Register 1 (PFCR1) controls the operation of Port p0 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for information on the masters.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC			0	RWSC				0	P0_DPFEN	0	P0_IPFEN	0	P0_PFLIM		P0_ BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

PFLASH_PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
8 P0_M7PFE	Port0 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P0_M4PFE	Port0 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–18 APC	Address Pipeline Control. This field controls the number of cycles that a subsequent flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1. For valid RWSC and APC combinations please refer to the device data sheet. **NOTE:** 1. A value of '1' in the most significant bit causes the flash controller to function in retrograde/legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the data sheet.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	000 Pipelined access to the flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 A pipelined access can be initiated 2 cycles before the previous data is valid 011 A pipelined access can be initiated 3 cycles before the previous data is valid 1xx Pipelined access to the flash is disabled and one wait state is inserted before a subsequent access can be initiated
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 RWSC	Read Wait State Control. This field controls the number of wait-states to be added to the best-case flash array access time for reads. The best-case flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the flash memory controller and the actual read access time of the flash memory controller. For valid RWSC and APC combinations please refer to the device data sheet. **NOTE:** 1. Higher operating frequencies require non-zero settings for this field for proper flash operation. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the device data sheet. This field applies to the configuration of Port0. 0000 No additional wait-states are added 0001 One additional wait-state is added ... 1111 Fifteen additional wait-states are added
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P0_DPFEN	Port0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P0_IPFEN	Port0 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P0_PFLIM	Port0 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BFEN_LK]=0. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

31.4.2 Platform Flash Configuration Register 3 (PFLASH_PFCR3)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_WCFG		0		0		0					Reserved	0			BFEN_LK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0											0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCR3 field descriptions

Field	Description
0–1 P0_WCFG	Port0 Way Configuration. This field controls the configuration of the line buffers for a given set acrosstwo ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers in an indexed set are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
	10 The buffers in an indexed set are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved.
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 BFEN_LK	BFEN Lock. This field controls access to the PFCR2[P0_BFEN] field, which enables or disables line read buffer hits and isare also used to invalidate the buffers. 0 The PFCR2[P0_BFEN] field is R/W. 1 PFCR2[P0_BFEN] field is Read-only.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.4.3 Platform Flash Access Protection Register (PFLASH_PFAPR)

The PFlash Access Protection Register (PFAPR) is used to control read and write accesses to the flash array.

NOTE

See the chip-specific platform flash controller information for details about the actual masters available on the device.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PFLASH_PFAPR field descriptions

Field	Description
0–1 M0AP	Master 0 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2–3 M1AP	Master 1 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4–5 M2AP	Master 2 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
6–7 M3AP	Master 3 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8–9 M4AP	Master 4 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10–11 M5AP	Master 5 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12–13 M6AP	Master 6 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
14–15 M7AP	Master 7 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16–17 M8AP	Master 8 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
18–19 M9AP	Master 9 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20–21 M10AP	Master 10 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22–23 M11AP	Master 11 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
24–25 M12AP	Master 12 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26–27 M13AP	Master 13 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
28–29 M14AP	Master 14 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30–31 M15AP	Master 15 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

31.4.4 Platform Flash Remap Control Register (PFLASH_PFCRCR)

The PFCRCR is used to globally enable/disable the calibration remap function.

CAUTION

Overlay remap functions, when used with line read buffers enabled (PFLASH_PFCRx[BFy_EN] field(s)='1') have a potential to cause data coherency issues, i.e., prefetched data in a line read buffer can become out-of-sync with data contained at the associated flash address range. To prevent this issue, always clear the flash controller line read buffers prior to enabling any remap region.

To clear the flash controller line read buffers, issue an interlock write command to a valid flash memory address. There will not be an actual flash program operation performed, but the flash controller mini-cache will be cleared as a side effect of executing the command. The interlock write must be issued from a core having appropriate access to flash. The sequence is as follows:

1. Set the C55FMC_MCR[PGM] field to '1'
2. Issue a 64-bit write to a valid flash address on a 64-bit boundary, i.e., the least significant 5 bits of the address are '0'
3. Clear C55FMC_MCR[PGM]

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							SAFE_CAL	0			IRMEN	0			GRMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRCR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SAFE_CAL	Safe Calibration. This bit facilitates the use of safety-critical calibration data. When this bit is enabled, the flash memory controller is configured to perform redundancy checking on the calibration remap function and reduce the total number of potential calibration regions from 8 to 4. This field is initialized by hardware reset. 1 Established calibration overlay regions are considered safety-critical. 0 Established calibration overlay regions are not considered safety-critical.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 IRMEN	Instruction Remap. This bit enables calibration remapping evaluation on instruction fetches. When PFCRCR[GRMEN] is disabled, this bit is ignored. This field is initialized by hardware reset. 0 Calibration remap evaluation is performed on any incoming data fetch requests only. 1 Calibration remap evaluation is performed on all incoming flash access requests - data and instruction fetches.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GRMEN	Global Remap Enable. This bit globally enables or disables the calibration remapping evaluation on system flash accesses. This field is initialized by hardware reset. 0 Calibration remap evaluation is not performed on any incoming system flash access requests. 1 Calibration remap evaluation is performed on all incoming system flash access requests.

31.4.5 Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)

The PFlash Calibration Remap Descriptor Enable Register (PFCRDE) is used to enable or disable up to 8 calibration remap descriptors. Note there is also a global remap enable (PFCRCR[GRMEN]) that also must be asserted in conjunction with the individual CRDnEN flags to enable a given remap descriptor.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRD0EN	CRD1EN	CRD2EN	CRD3EN	CRD4EN	CRD5EN	CRD6EN	CRD7EN	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRDE field descriptions

Field	Description
0 CRD0EN	Calibration Remap Descriptor 0 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 0 invalid 1 Calibration Remap Descriptor 0 valid
1 CRD1EN	Calibration Remap Descriptor 1 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 1 invalid 1 Calibration Remap Descriptor 1 valid
2 CRD2EN	Calibration Remap Descriptor 2 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset.

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	0 Calibration Remap Descriptor 2 invalid 1 Calibration Remap Descriptor 2 valid
3 CRD3EN	Calibration Remap Descriptor 3 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 3 invalid 1 Calibration Remap Descriptor 3 valid
4 CRD4EN	Calibration Remap Descriptor 4 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 4 invalid 1 Calibration Remap Descriptor 4 valid
5 CRD5EN	Calibration Remap Descriptor 5 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 5 invalid 1 Calibration Remap Descriptor 5 valid
6 CRD6EN	Calibration Remap Descriptor 6 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 6 invalid 1 Calibration Remap Descriptor 6 valid
7 CRD7EN	Calibration Remap Descriptor 7 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 7 invalid 1 Calibration Remap Descriptor 7 valid
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.4.6 Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD*n*_Word0)

Each 96-bit (12-byte) region descriptor specifies an overlay region where a flash access can be remapped during calibration and debug. The calibration remap descriptors are organized sequentially as 128-bit (16-byte) structures in the Platform Flash Controller's programming model. Each of the three 32-bit words that define a single calibration region are detailed in the subsequent sections; the fourth word is unused.

The first word of the flash memory controller overlay region descriptor defines the 0-modulo-size logical start (byte) address of the calibration remap region. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

The hardware performs basic checks on the validity of the logical start address when this word is written; if a non-supported logical start address is defined, the register write is terminated with an error and the register left unchanged and the valid bit cleared. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 100h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LSTARTADDR																												0			
W																													▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word0 field descriptions

Field	Description
0–27 LSTARTADDR	Logical Start Address. This field defines the most significant bits of the 0-modulo-size logical start address of the overlay remap region. It corresponds to the logical system address which maps to the flash memory space. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.4.7 Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD*n*_Word1)

The second word of the flash memory controller calibration region descriptor defines the 0-modulo-size byte physical start (byte) address of the calibration remap region.

The contents of this word define the targeted destination overlay memory. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Attempted writes to this register with illegal values, as defined by the sizes and locations of the overlay memories, are terminated with an error and clear the valid bit. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 104h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSTARTADDR																												0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word1 field descriptions

Field	Description
0–27 PSTARTADDR	Calibration Remap Descriptor *n* Physical Start Address - This field defines the most significant bits of the 0-modulo-size physical start byte address of the calibration remap descriptor. This address corresponds to the physical address which maps to the destination calibration overlay memory. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.4.8 Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD*n*_Word2)

The third word of the calibration region descriptor defines a per-master calibration remap enable and the remap region size. For cacheable spaces being remapped, the minimum region size is 32 bytes to match the flash page and cache line sizes. Writes to this word clear the calibration remap descriptor's valid bit.

Address: 0h base + 108h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0EN	M1EN	M2EN	M3EN	M4EN	M5EN	M6EN	M7EN	M8EN	M9EN	M10EN	M11EN	M12EN	M13EN	M14EN	M15EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											CRDSize				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word2 field descriptions

Field	Description
0 M0EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 0 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 0 as defined by this region descriptor.
1 M1EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 1 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 1 as defined by this region descriptor.
2 M2EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 2 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 2 as defined by this region descriptor.
3 M3EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 3 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 3 as defined by this region descriptor.
4 M4EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 4 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 4 as defined by this region descriptor.
5 M5EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 5 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 5 as defined by this region descriptor.
6 M6EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 6 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 6 as defined by this region descriptor.
7 M7EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 7 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 7 as defined by this region descriptor.
8 M8EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 8 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 8 as defined by this region descriptor.
9 M9EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 9 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 9 as defined by this region descriptor.
10 M10EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 10 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 10 as defined by this region descriptor.
11 M11EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 11 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 11 as defined by this region descriptor.
12 M12EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 12 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 12 as defined by this region descriptor.
13 M13EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 13 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 13 as defined by this region descriptor.
14 M14EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 14 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 14 as defined by this region descriptor.
15 M15EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 15 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 15 as defined by this region descriptor.
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 CRDSize	Calibration Remap Descriptor *n* Size - This field specifies the size of the calibration remap region. 00000 Reserved 00001 Reserved 00010 Reserved 00011 Reserved 00100 Reserved 00101 Region size is 32 bytes. Low-order 5 bits of LSTARTADDR and PSTARTADDR must be zero 00110 Region size is 64 bytes. Low-order 6 bits of LSTARTADDR and PSTARTADDR must be zero 00111 Region size is 128 bytes. Low-order 7 bits of LSTARTADDR and PSTARTADDR must be zero 01000 Region size is 256 bytes. Low-order 8 bits of LSTARTADDR and PSTARTADDR must be zero 01001 Region size is 512 bytes. Low-order 9 bits of LSTARTADDR and PSTARTADDR must be zero 01010 Region size is 1 KB. Low-order 10 bits of LSTARTADDR and PSTARTADDR must be zero 01011 Region size is 2 KB. Low-order 11 bits of LSTARTADDR and PSTARTADDR must be zero

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	01100 Region size is 4 KB. Low-order 12 bits of LSTARTADDR and PSTARTADDR must be zero 01101 Region size is 8 KB. Low-order 13 bits of LSTARTADDR and PSTARTADDR must be zero 01110 Region size is 16 KB. Low-order 14 bits of LSTARTADDR and PSTARTADDR must be zero 01111 Region size is 32 KB. Low-order 15 bits of LSTARTADDR and PSTARTADDR must be zero 10000 Region size is 64 KB. Low-order 16 bits of LSTARTADDR and PSTARTADDR must be zero 10001 Region size is 128 KB. Low-order 17 bits of LSTARTADDR and PSTARTADDR must be zero 10010 Region size is 256 KB. Low-order 18 bits of LSTARTADDR and PSTARTADDR must be zero 10011 Region size is 512 KB. Low-order 19 bits of LSTARTADDR and PSTARTADDR must be zero 10100 Region size is 1 MB. Low-order 20 bits of LSTARTADDR and PSTARTADDR must be zero 10101 Region size is 2 MB. Low-order 21 bits of LSTARTADDR and PSTARTADDR must be zero 10110 Region size is 4 MB. Low-order 22 bits of LSTARTADDR and PSTARTADDR must be zero 10111 Region size is 8 MB. Low-order 23 bits of LSTARTADDR and PSTARTADDR must be zero 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

31.5 Functional description

The flash memory controller interfaces between:

- The AHB system bus port
- Flash memory array
- System RAM

For accesses targeting the flash array, the flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the flash array. The flash memory controller captures read data from the flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of two ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

31.5.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Thus, the access time of the operation is determined by the setting of the PFCR1[RWSC] field. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

31.5.2 Access protections

On-chip Flash memory accesses are subject to restrictions based on chip security implementation and/or requirements based on functional restrictions of the flash memory, e.g., a read access to a flash block while it is currently being written is not allowed. When a prohibited access is attempted, a system bus error termination results.

The flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding read access control or write access control settings do not allow the access, thus causing a protection violation. See Platform Flash Access Protection Register (PFLASH_PFAPR) for more detail. In this case the flash memory controller does not initiate the requested flash array access.
- Attempted access by an AHB master to a flash region where the corresponding censorship control is asserted.
- The flash returns an error response on an attempted access to a partition that is unavailable. (see Flash error response operation or Embedded Flash Memory).
- Attempted access by an AHB master to a reserved region in the flash memory map.

The flash array may also signal an error response to terminate a requested access due to improper sequencing during program/erase operations, improper sequencing during array integrity testing. When an error response is received the flash memory controller does not update or validate a page read buffer. An error response may be signaled on a read or interlock write operation. For more information on the specifics related to signaling of flash errors, including flash ECC events, array integrity testing and read-while-write events, refer to the flash memory chapter.

31.5.2.1 Censorship

The flash space is defined by regions:

- Code flash (instruction and constant data)
- Data flash (data for EEPROM emulation)
- UTEST flash

Each section can be independently censored, and the flash memory controller provides independent censorship control inputs for each region on a read vs. write basis.

31.5.2.2 PFAPR - Platform Flash Access Protection Register (PFAPR)

The flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in Platform Flash Access Protection Register (PFLASH_PFAPR). Detection of a protection violation results in an error response from the flash memory controller on the AHB transfer.

31.5.3 Access pipelining

Accesses to the flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by the AHB master. Access pipelining may be applied only to read cycles targeting the flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal flash access and is not supported for calibration overlay RAM fetches. On calibration reads, the setting of PFCR1[APC] is ignored.

NOTE

> Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.

31.5.4 Line read buffers and prefetch operation

The AHB ports of the flash memory controller contains a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see Platform Flash Configuration Register 1 (PFLASH_PFCR1)). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32-bytes, and a subsequent flash access is initiated. A flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

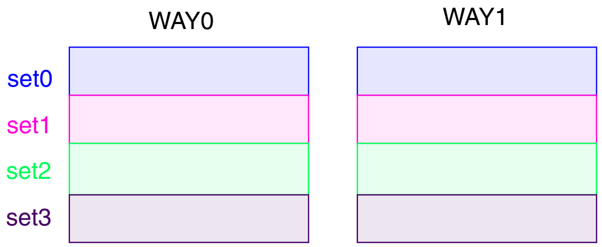


Figure 31-2. Flash memory controller 4-entry, 2-way mini-cache organization

Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in Platform Flash Configuration Register 1 (PFLASH_PFCR1). More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR1[P0_BFEN] must be set to ‘1’,

2. PFCR1[P0_PFLIM] must be non-zero, and
3. Either PFCR1[P0_IPFEN] or PFCR1[P0_DPFEN] must be asserted.

Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

31.5.5 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls. Prefetches are never triggered by write cycles.

31.5.6 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

31.5.7 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration (P0_WCFG) field. Refer to Platform Flash Configuration Register 3 (PFLASH_PFCR3). The buffers can be organized as a "pool" of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, way 0 is allocated for instruction fetches and way 1 for data accesses.

31.5.8 PFlash calibration remap support

The flash memory controller supports calibration development by providing a remapping function to route flash accesses to on-chip System RAM that can be used as overlay RAM.

The overlay function supports the following features:

- Can be mapped over internal flash memory
    - Allows calibration of constant data without requirement for additional external RAMs and calibration memory interfaces

- Protected with error detection and correction mechanism
- Accesses to all overlay RAMs can achieve the same timing as accesses to internal flash
    - Timing for calibration accesses to overlay RAM is only guaranteed if RWSC is set sufficiently high to cover the intrinsic burst access incurred when fetching a full page of calibration data
- Support for up to 32 distinct calibration remap regions.
- Protection against accidental enable and reconfiguration of calibration remap function :
    - Two stage mechanism to enable remap, using both global and individual calibration region descriptor enable bits
    - Optional configuration of calibration region descriptors as redundant pairs, allowing detection of accidental region reconfiguration.
- Support for overlay remapping to unique calibration regions on a per-master basis.

The calibration remap function supports the following overlay targets:

- A portion of the system RAM can be used for overlay.

31.5.8.1 PFlash calibration remap to RAM

The overlay RAM is selected based on the translated physical address.

The translated physical address defines the destination memory. Table 31-1 shows the base addresses for the calibration data memories as defined by the system memory map.

Table 31-1. Calibration data memory base addresses

Calibration Memory	Base Address	Possible Access Sources
Flash Arrays	0x0{8,9,A}--_----	P0, P1
System RAM	0x40{0,1}-_----	P0, P1

The system RAM connection from the calibration remap logic is implemented via a private 64-bit connection to the platform RAM controller.

Accesses to the flash arrays always use the logical, non-remapped system address. Calibration data store references to the system RAM always use the physical, non-remapped system address. Conversely, calibration data load references can be accessed using a translated physical address or, if provided by the requesting master, the physical, non-remapped system address.

The read data interface of system RAM is 64 bits wide , whereas the flash read data interface is 256 bits wide. As a result, a remapped calibration load access initiates a four-beat burst sequence to mimic the throughput of a single flash access. There is a direct relationship between the operating frequency range and the required number of wait states to correctly sample flash read data. By contrast, a remapped calibration load access has a fixed, intrinsic 4-beat latency. Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. Expressed mathematically:

$$(\mathrm{RWSC}+2)\geq\left(\frac{\mathrm{IntrinsicFlashAccessTime}}{\mathrm{OperatingFrequencyClockPeriod}}+2\right)\geq(\mathrm{OverlayRAMTransferBurstLatency})$$

Equation 15. Access time

Notice the overall latency on a random, initial flash access that does not hit in the prefetch mini-buffer incurs an additional cycle latency beyond the intrinsic flash access time. This cycle accounts for pipeline stalls incurred by the flash controller state machine. The formula above is based on specified intrinsic flash access times of the C55FMC array at 150 °C. Calibration data that is loaded in the flash controller's mini-cache is returned with single-cycle latency.

The memory-mapped calibration remap (region) descriptor (CRD*n*) registers define the required address translation. Each calibration remap descriptor includes the logical base address, the translated physical base address and a region size (plus other control bits). The calibration remap evaluation is initiated when the flash controller is presented with a flash access request from any system bus master to a location in the mirrored flash address space. Conversely, references using the associated non-mirrored flash system address are not evaluated for calibration remapping.

The address translation mechanism requires the logical and physical base addresses of the region be aligned on 0-modulo-size boundaries.

The region size is decoded to create the appropriate address bit enables which are then applied to the registered logical flash address. An equality comparator then compares the adjusted logical flash address against the logical address from the calibration remap descriptor to determine the region "hit". The hit determination is further qualified based on the bus master number that initiated the flash access. At the same time, the logical flash address and the physical address defined in the calibration remap descriptor are merged to form the translated address to be used in the event of a calibration region hit. Aside from the individual instantiations of the calibration remap descriptors, this logic examines all the descriptor hit indicators to select the appropriate translated address.

In the absence of a region "hit" data is returned from the flash at the provided logical address.

It should be noted in the event of *overlapping remap regions*, the calibration remap descriptor with the largest number is used for the address translation, that is, if a logical flash address hits in multiple descriptors, say CRDx and CRDy, the translated address from CRDy is used, given y > x.

In general, the calibration remap logic operates only on flash *data* accesses. However, flash instruction references can optionally remapped while in debug modes of operation to support the use of software breakpoints when PFCRC[IRMEN] is set.

31.5.9 Reliability considerations

This section summarizes mechanism to enhance the reliability of flash by correcting and detecting faults.

31.5.9.1 e2eECC End-to-End ECC

31.5.9.1.1 e2eECC and flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with flash memory. The basic issues involve: the default state of a flash block that is erased (all ones) and the fact that programming an erased flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the flash memory in three ways.

First, the all-ones codeword, since it reflects the state of an erased flash location, is treated as a valid error free encoding; in particular, it is required that the checkbits associated with the all all-ones value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased flash block generates an all-ones data for all locations, the address field cannot be included in the ECC code used by the flash. The flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the flash memory and its support for EEPROM emulation. Lastly, there are ECC implications associated with the storage and retrieval of overlaid calibration data.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the flash memory in multiple ways. The required ECC adjustments are handled by the platform flash controller before data is written to the flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all-ones erased state.

On a flash read operation, a similar but "reversed" set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following flash ECC example. Let the flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns fl_rdata = 0xFFFFFFFF_FFFFFFFF, fl_cdata = 0xFF. The flash memory controller calculates the addr_chkbit = 0xC4 using the H-matrix; it inverts the fl_cdata vector to 0x00 and factors in the addr_chkbit = 0xC4 to produce the following valid e2eECC codeword:

   addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

   addr = 0x00345678, wdata = 0xBABEFACE_DEADBEEF, wchkbit = 0xE3

   Before performing the interlock write to the flash array, the flash memory controller adjusts the chkbit value as fl_wchkbit = 0xD8 (0xE3 ^ 0xff ^ 0xC4) by toggling the original write checkbits and then factoring in the addr_ecc. Accordingly, the codeword stored in the flash array is:

   fl_data = 0xBABEFACE_DEADBEEF, fl_checkbit = 0xD8

3. Read address 0x00345678. The flash array returns the stored codeword, fl_rdata = 0xBABEFACE_DEADBEEF, fl_rchkbit = 0xD8. The flash memory controller inverts the checkbit vector and factors in the address checkbits (0xD8 ^ 0xFF ^ 0xC4) to produce:

   addr = 0x00345678, rdata = 0xBABEFACE_DEADBEEF, rchkbit = 0xE3

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The flash memory controller performs EDC after ECC check on all flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

31.5.9.2 Flash address generation check

Fault detection coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Fault detection coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. Recall on a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash interface signals to invoke a flash lookup. In addition to providing the requested read data, the flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.

This sideband information is used by the flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

31.5.9.2.1 Overlay RAM feedback check

The integrity of the address and data on an overlay RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the flash memory controller is handled by a transaction monitor which verifies the integrity of the transactions between the flash memory controller and the on-chip overlay RAM. The function of the transaction monitor relies on a feedback path between the flash memory controller and the on-chip overlay RAM, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated. This feedback information is used by the flash memory controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and on-chip overlay RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

31.5.9.2.2 Reliability considerations on overlay accesses

Because calibration write accesses are always presented with the physical, non-remapped system address, the write data checkbits presented by the master are calculated based on the physical overlay RAM system address. On a subsequent overlay read, the ECC checkbits must be manipulated to replace the physical overlay RAM address contribution with the logical flash address contribution. This is required to satisfy the e2eECC check at the originating master.

In addition, the flash memory controller can be configured to treat calibration RAM space as a safety-critical component. By programming PFCRCR[SAFE_CAL]=1, the flash memory controller treats the set of available calibration region descriptors as a replicated pair, with each set containing half the total number of available calibration region descriptors.

If there are 32 calibration region descriptors, CRD0 – CRD31, and PFCRCR[SAFE_CAL]=1, the flash memory controller treats CRD0 – CRD15 and CRD16 – 32 as replicated sets of 16 calibration region descriptors, where:

- CRD0 and CRD16 are configured identically
- CRD1 and CRD17 are configured identically
- CRD2 and CRD18 are configured identically
- ...
- and CRD15 and CRD31 are configured identically

Expressed more generally, the contents of CRD[*n*] must be identical to the contents of CRD[(N/2) + *n*], where N is the total number of region descriptors and 0 ≤ *n* ≤ (N/2) – 1.

Note

> It is the user's responsibility to establish redundant calibration region descriptors by programming the associated pairs of CRDs.

Incoming flash read requests are compared simultaneously against the calibration regions defined in CRD0 to CRD[(N/2) – 1] and the calibration regions defined in CRD[N/2] to CRD[N – 1]. Due to the enabled redundancy, the parallel overlay remap hardware structures should evaluate to symmetric results. If a mismatch is detected, the event is reported to the FCCU and calibration remapping is not performed. Instead, data is returned from the flash.

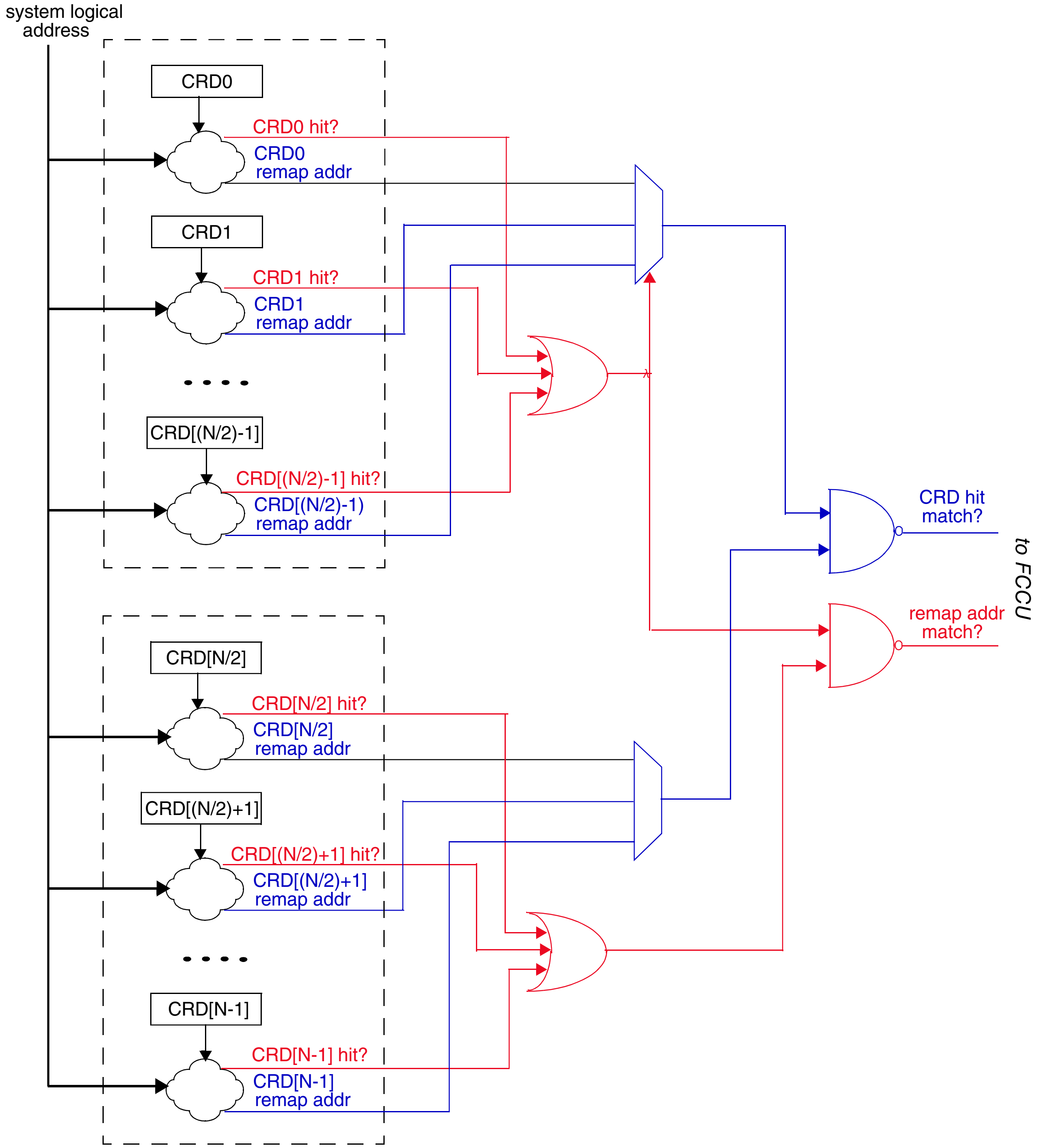


Figure 31-3. Safety-critical calibration remap datapath with redundancy

31.5.10 ECC on data flash accesses

ECC events detected on accesses to data flash blocks are suppressed from being reported to the Memory Error Management Unit (MEMU).

In the event of a single-bit correction, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is suppressed from being reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode value is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address, and the non-correctable error event is suppressed from being reported to the MEMU. For non-correctable error detection, the flash memory controller returns a value of FFFF_FFFFh to the requesting master.

NOTE

EEPROM should be avoided for storage of executable code.

31.5.11 Array integrity considerations

During an array integrity sequence, the flash memory array ignores any incoming read requests. When a flash array integrity check is in progress, the flash memory controller terminates all flash access requests with an error. More specifically, it aborts the incoming flash access requests and terminates the system bus transfer with an error.

Chapter 32
Embedded Flash Memory (c55fmc)

32.1 Introduction

The primary function of the embedded flash memory is to serve as electrically programmable and erasable non-volatile memory (NVM) that may be used for instruction or data storage.

The following figure shows the top-level diagram and functional organization of the flash memory unit.

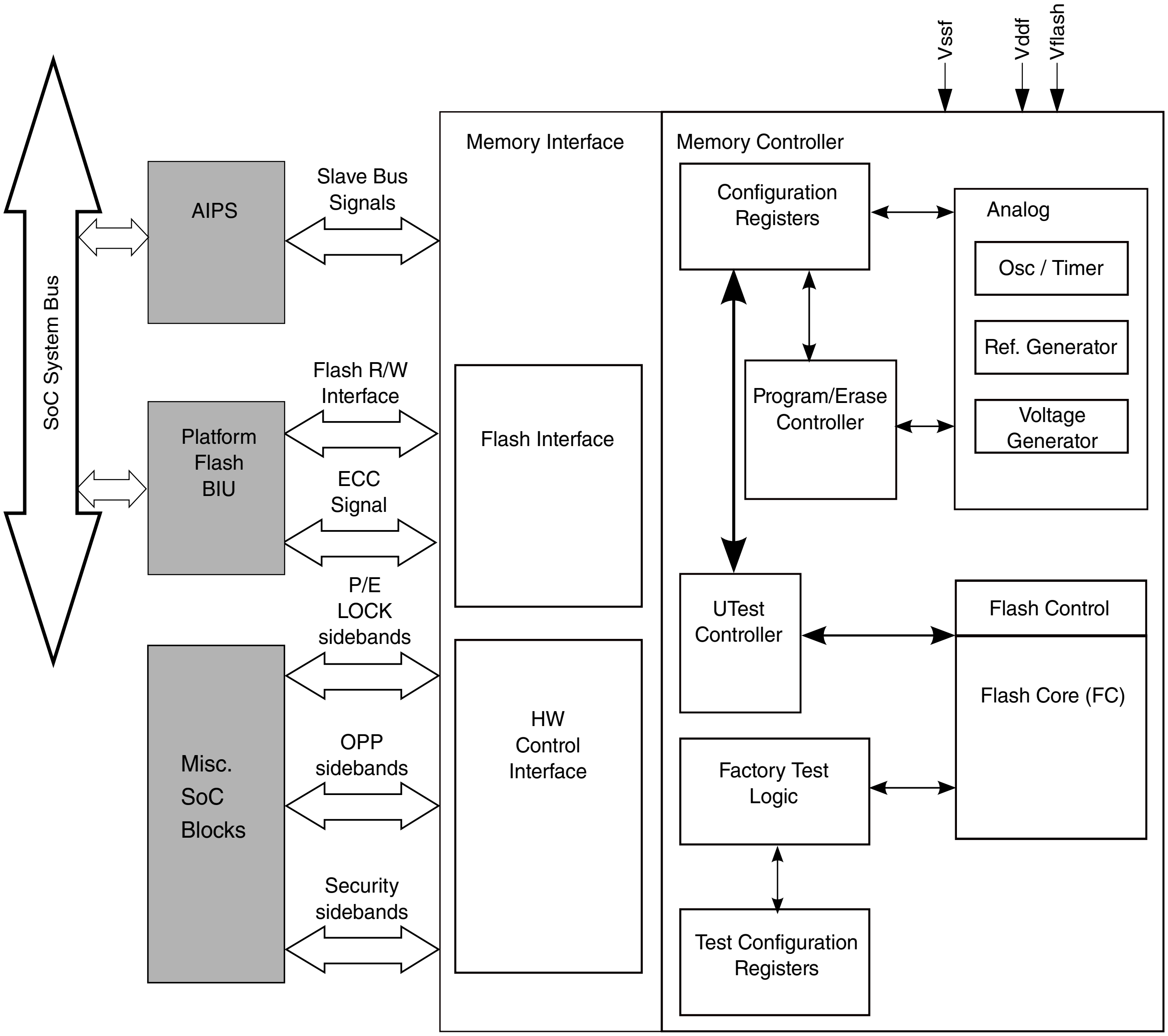


Figure 32-1. c55fmc block diagram

32.1.1 Overview

The embedded flash memory is addressable by word (32 bits) or double word (64 bits) for program operation, and page (256 bits) for read operation. Multiple word or double-word writes may be done to the flash memory to fill up the program page buffer (256 bits), enabling page programming (256 bits, requiring 4 double-word writes) and quad-page programming (1024 bits, requiring 16 double-word writes). Flash memory reads always return 256 bits, although read page buffering may be performed by the Bus Interface Unit (BIU).

For more details on the embedded flash memory architecture and features, see Functional Description.

32.1.2 Features

The embedded flash memory includes these distinct features:

- Test information stored in a dedicated nonvolatile block (referred to as the UTest block)
- OTP space made available in the UTest block
- Read page size of 256 bits (8 words)
- ECC with single-bit correction, double-bit detection (all 1's valid)
- Quad Page programming (64-bit granularity)
- Software programmable block program/erase restriction control
- Erasing of selected block(s)
- Independent programming of the UTest NVM block
- Embedded hardware program and erase algorithms
- Support for reading while writing when the accesses are to different partitions
- Erase suspend, program suspend, and erase-suspended program
- UTest mode (user-accessible test modes) including Array Integrity and Margin Read
- Triple Voted Flops for flash functions requiring high reliability, e.g., internal trimming, redundancy, and mode control

32.1.3 Modes of operation

Following is a brief description of the embedded flash memory operating modes.

- *User mode* is the default operating mode of the embedded flash memory. In this mode, it is possible to read and write registers, read and interlock write the memory array, program the memory array, and erase the memory array. In this mode program and erase operations are initiated by doing array and register writes, and are controlled by an internal state machine.
- *Low power mode* turns off all DC current sources within the embedded flash memory on the Vflash domain, and enables VDDF to be power gated. The embedded flash memory is not accessible for read, write, program, or erase when in low power mode.
- *UTest mode* is a tiered test mode strategy in which a portion of the factory test modes are made available. This mode is protected but accessible.

32.2 UTest NVM block

The UTest NVM block may be enabled by the BIU. When the UTest NVM space is enabled, all operations are mapped to the UTest NVM block. User-mode programming of the UTest NVM block is enabled only when MCR[PEAS] is high. The UTest NVM block is an OTP block - thus erase is not allowed.

The UTest NVM block supports RWW, and is grouped with the blocks in partition 0.

The UTest NVM block may be locked against program by using the lock registers.

Programming of the UTest NVM space has restrictions that are similar to those of the main space in terms of how ECC is calculated. Only one program operation is allowed per 64-bit ECC segment.

The UTest NVM block contains specified data that is needed for embedded flash memory or SoC features.

32.3 Memory map and register definition

The embedded flash memory map consists of a flash memory array (which includes main array space and UTest NVM space) and a region of registers associated with the programming model that enable flash memory array operation and modification.

The address space consists of 16 KB, 32 KB, 64 KB, and 256 KB blocks..

C55FMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (C55FMC_MCR)	32	R/W	0000_0600h	32.3.1/900
8	Extended Module Configuration Register (C55FMC_MCRE)	32	R	See section	32.3.2/905
10	Lock 0 register (C55FMC_LOCK0)	32	R/W	BFFF_FFFFh	32.3.3/908
14	Lock 1 register (C55FMC_LOCK1)	32	R/W	0000_FFFFh	32.3.4/910
18	Lock 2 register (C55FMC_LOCK2)	32	R/W	FFFF_FFFFh	32.3.5/910
1C	Lock 3 register (C55FMC_LOCK3)	32	R/W	0000_FFFFh	32.3.6/911
38	Select 0 register (C55FMC_SEL0)	32	R/W	0000_0000h	32.3.7/912
3C	Select 1 register (C55FMC_SEL1)	32	R/W	0000_0000h	32.3.8/913
40	Select 2 register (C55FMC_SEL2)	32	R/W	0000_0000h	32.3.9/914

Table continues on the next page...

C55FMC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
44	Select 3 register (C55FMC_SEL3)	32	R/W	0000_0000h	32.3.10/ 915
50	Address register (C55FMC_ADR)	32	R	0000_0000h	32.3.11/ 916
54	UTest 0 register (C55FMC_UT0)	32	R/W	0000_0001h	32.3.12/ 918
58	UMISR register (C55FMC_UM0)	32	R/W	0000_0000h	32.3.13/ 921
5C	UMISR register (C55FMC_UM1)	32	R/W	0000_0000h	32.3.13/ 921
60	UMISR register (C55FMC_UM2)	32	R/W	0000_0000h	32.3.13/ 921
64	UMISR register (C55FMC_UM3)	32	R/W	0000_0000h	32.3.13/ 921
68	UMISR register (C55FMC_UM4)	32	R/W	0000_0000h	32.3.13/ 921
6C	UMISR register (C55FMC_UM5)	32	R/W	0000_0000h	32.3.13/ 921
70	UMISR register (C55FMC_UM6)	32	R/W	0000_0000h	32.3.13/ 921
74	UMISR register (C55FMC_UM7)	32	R/W	0000_0000h	32.3.13/ 921
78	UMISR register (C55FMC_UM8)	32	R/W	0000_0000h	32.3.13/ 921
7C	UMISR register (C55FMC_UM9)	32	R/W	0000_0000h	32.3.14/ 922
80	Over-Program Protection 0 register (C55FMC_OPP0)	32	R	See section	32.3.15/ 923
84	Over-Program Protection 1 register (C55FMC_OPP1)	32	R	See section	32.3.16/ 924
88	Over-Program Protection 2 register (C55FMC_OPP2)	32	R	See section	32.3.17/ 925
8C	Over-Program Protection 3 register (C55FMC_OPP3)	32	R	See section	32.3.18/ 925

32.3.1 Module Configuration Register (C55FMC_MCR)

NOTE

1. A number of Module Configuration Register (MCR) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in Functional Description.
2. See Functional Description for information about simultaneous MCR writes, and priority levels.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RVE	RRE	AEE	EEE	0											
W	w1c	w1c	w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	0			PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

C55FMC_MCR field descriptions

Field	Description
0 RVE	Read Voltage Error RVE provides information on previous reads monitoring the read pump voltage. If the read voltage is detected to be out of range, this bit is set to indicate that previous reads requested may have been

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without voltage issues 1 A previous read may have been corrupted due to read voltage being out of range
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the read reference. If the read reference is detected to be out of range, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without reference issues 1 A previous read may have been corrupted due to read reference being out of range
2 AEE	Address Encode Error AEE provides information on previous reads monitoring the address encode feature. On every read request to the flash, the incoming address is compared to an encoded address (row, column, and block) coming back from the memory array using the read data sense amplifier timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error will be recorded. If an address encode mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without address encode mismatches 1 A previous read may be corrupted based on address encode mismatch
3 EEE	ECC after ECC Error EEE provides information on previous reads monitoring the ECC after ECC feature. On every read request to the flash, ECC is recalculated serially, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error will be reported. If an ECC after ECC mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without ECC after ECC mismatches 1 A previous read may be corrupted based on ECC calculation errors
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EER	ECC Event Error This bit provides information on previous reads. If a double bit detection occurred, the EER bit is set to a '1'. This bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be set by software. In the event of a single bit detection and correction, this bit is not set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 An ECC Error occurred during a previous read.
17 RWE	Read-While-Write Event Error This bit provides information on previous read-while-write (RWW) reads. If an RWW error occurs, this bit is set to 1. The bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be written to a 1. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 A read-while-write error occurred during a previous read.
18 SBC	Single Bit Correction SBC provides information on previous reads, if the SBCE is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without corrections. 1 A Single Bit Correction occurred during a previous read.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 PEAS	Program Access Space PEAS indicates which space is valid for program or erase operations — either main array space or UTest NVM space. It should be noted that erase is not allowed on the UTest NVM space. PEAS = 0 indicates the main address space is active for all program or erase operations. PEAS = 1 indicates the UTest NVM address space is active for program or erase (if applicable) operations. The value in PEAS is captured and held when the UTest NVM block is enabled with the first interlock write done for program operations. The value of PEAS is retained between sampling events (in other words, subsequent first interlock writes). The value in PEAS may be changed during erase-suspended program, and reverts back to its original state once the erase-suspended program is completed. PEAS is read only. 0 UTest NVM address space is disabled for program/erase and main address space enabled. 1 UTest NVM address space is enabled for program/erase and main address space disabled.
21 DONE	State Machine Status DONE indicates whether the embedded flash memory is performing a high voltage operation. DONE is set to a 1 on termination of the embedded flash memory reset. DONE is read only. DONE is set to a 1 at the end of program and erase high voltage sequences. **NOTE:** This bit transitions from a 0 to 1 during reset and remains at 1 after reset. 0 Flash memory is executing a high voltage operation. 1 Flash memory is not executing a high voltage operation.
22 PEG	Program/Erase Good The PEG bit indicates the completion status of the last flash memory program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the embedded flash memory is reset. PEG is read only. The value of PEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	PEG will be 0 if an Erase-Suspended Program is attempted to a block that has been selected for Erase. PEG will also be 0 in the event of an error in the interlock write for program only. An error in the interlock write occurs when the ECC value provided to the flash memory does not match the ECC calculated within the flash memory. In the event of this, high voltage will not be applied to the array, and PEG will indicate a failed program. **NOTE:** 1. If program or erase operations are attempted on blocks that are locked, the response from embedded flash memory is PEG = 1, indicating that the operation was successful, and the contents of the block are properly protected from the program or erase operation. PEG = 1 is also true if an abort occurs during an HV request to a locked block. 2. If a program operation is attempted to a location marked as OTP, the response from the embedded flash memory is PEG = 0, indicating that the operation was not allowed, and the value interlocked was not programmed, since the desired doubleword was already programmed with a previous program operation. 0 Program or erase operation failed. 1 Program or erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag due to a high voltage event (program or erase) finishing (normal, abort, suspend). If PECIE is written while not in a high voltage event, the interrupt will not immediately trigger, but will trigger after the next high voltage event is completed. 0 An interrupt request will not be generated when the DONE flag is set. 1 An interrupt request will be generated when the DONE flag is set.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PGM	Program PGM is used to set up embedded flash memory for a program operation. A 0 to 1 transition of PGM initiates a program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions: • User mode read (ERS is low and UTE is low) • Erase suspend (ERS and ESUS are 1) with EHV low PGM can be cleared only when PSUS and EHV are low and DONE is high. PGM is cleared on reset. 0 Flash memory is not executing a program sequence. 1 Flash memory is executing a program sequence.
28 PSUS	Program Suspend PSUS is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state. The embedded flash memory is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the embedded flash memory in program suspend. The embedded flash memory enters suspend within Tpsus of this transition. PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to program. The embedded flash memory cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
29 ERS	Erase ERS is used to set up embedded flash memory for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set in user mode read (PGM is low and UTE is low). ERS can be cleared only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash memory is not executing an erase sequence. 1 Flash memory is executing an erase sequence.
30 ESUS	Erase Suspend ESUS is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state. The embedded flash memory is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in erase suspend. The embedded flash memory enters suspend within Tesus of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to erase. The embedded flash memory cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31 EHV	Enable High Voltage The EHV bit enables the embedded flash memory for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock under one of the following conditions: • Erase (ERS = 1, ESUS = 0) • Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0) • Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0) If a program operation is to be initiated while an erase is suspended, EHV must be cleared while in erase suspend before setting PGM. In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS low, and ESUS low terminates the current program/erase high-voltage operation. In an erase-suspended program operation, a 1 to 0 transition of EHV with DONE high, ESUS high, PGM high, ERS high, and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. EHV may not be written to a 1 after an abort is requested (EHV being cleared) and before DONE transitions high. A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the embedded flash memory to exit suspend. EHV may not be written after a suspend bit is set high and before DONE transitions high. EHV may not be set low after the current suspend bit is set low and before DONE transitions low. **NOTE:** Aborting a high voltage operation leaves FC addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks. 0 Flash memory is not enabled to perform a high voltage operation. 1 Flash memory is enabled to perform a high voltage operation.

32.3.2 Extended Module Configuration Register (C55FMC_MCRE)

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HT	0		n256K					n64Kh			n32Kh		n16Kh		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	n64Km			n32Km		n16Km			n64Kl			n32Kl		n16Kl		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_MCRE field descriptions

Field	Description
0 HT	High Temperature Enabled. HT provides configuration information of the module. If the flash is tuned for high temperature operation (165 °C) at reduced performance (slower read at 165 °C and less and slower program and erase at 150 °C or less), and limited features at this temperature (no margin reads), the HT status bit will be a 1. Default is 150 °C as maximum operating temperature, and the HT status bit will be a 0. HT is read only. 0 150 °C maximum operating temperature. 1 165 °C maximum operating temperature with reduced performance.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 n256K	Number of 256 KB blocks. The value of this field is dependent upon the size of the embedded flash memory. This field is read only. 00000 Zero 256 KB blocks. 00001 Two 256 KB blocks (512 KB total). 00010 Four 256 KB blocks (1 MB total). 00011 Six 256 KB blocks (1.5 MB total). 00100 Eight 256 KB blocks (2 MB total). 00101 Ten 256 KB blocks (2.5 MB total).

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	00110 Twelve 256 KB blocks (3 MB total). 00111 Fourteen 256 KB blocks (3.5 MB total). 01000 Sixteen 256 KB blocks (4 MB total). 01001 Eighteen 256 KB blocks (4.5 MB total). 01010 Twenty 256 KB blocks (5 MB total). 01011 Twenty-two 256 KB blocks (5.5 MB total). 01100 Twenty-four 256 KB blocks (6 MB total). 01101 Twenty-six 256 KB blocks (6.5 MB total). 01110 Twenty-eight 256 KB blocks (7 MB total). 01111 Thirty 256 KB blocks (7.5 MB total). 10000 Thirty-two 256 KB blocks (8 MB total). 10001 Reserved 10010 Reserved 10011 Reserved 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Forty-eight 256 KB blocks (12 MB total). 11001 Reserved. 11010 Reserved. 11011 Reserved. 11100 Reserved. 11101 Reserved. 11110 Reserved. 11111 Reserved.
8–10 n64Kh	Number of 64 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Twelve 64 KB blocks.. 110 Sixteen 64 KB blocks. 111 Reserved.
11–12 n32Kh	Number of 32 KB blocks in partitions 4 and 5. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13–15 n16Kh	Number of 16 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks.

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16–18 n64Km	Number of 64 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19–20 n32Km	Number of 32 KB blocks in partitions 2 and 3. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21–23 n16Km	Number of 16 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
24–26 n64Kl	Number of 64 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
27–28 n32Kl	Number of 32 KB blocks in partitions 0 and 1. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
29–31 n16Kl	Number of 16 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Reserved. 101 Reserved. 110 Reserved. 111 Reserved.

32.3.3 Lock 0 register (C55FMC_LOCK0)

The Lock 0 (LOCK0) register provides a means to protect blocks from being modified.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCK	0	LOWLOCK													
W		▨														
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCK															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK0 field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCK register signifies that the UTest NVM block is locked for program. A value of 0 in the TSLOCK register signifies that the UTest NVM block is available to receive program pulses. The TSLOCK register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, TSLOCK register is not writable if a high voltage operation is suspended.

Table continues on the next page...

C55FMC_LOCK0 field descriptions (continued)

Field	Description
	0 UTest NVM block is available to be programmed. 1 UTest NVM block is locked and cannot be programmed.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations
16–31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

32.3.4 Lock 1 register (C55FMC_LOCK1)

The Lock 1 (LOCK1) register provides a means to protect blocks from being modified.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

32.3.5 Lock 2 register (C55FMC_LOCK2)

The Lock 2 (LOCK2) register provides a means to protect blocks from being modified.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KLOCK																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK2 field descriptions

Field	Description
0–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

32.3.6 Lock 3 register (C55FMC_LOCK3)

The Lock 3 (LOCK3) register provides a means to protect blocks from being modified.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A256KLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted.

Table continues on the next page...

C55FMC_LOCK3 field descriptions (continued)

Field	Description
	The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

32.3.7 Select 0 register (C55FMC_SEL0)

The Select 0 (SEL0) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWSEL														MIDSEL															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWSEL	LOW Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. LOWSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

Table continues on the next page...

C55FMC_SEL0 field descriptions (continued)

Field	Description
16–31 MIDSEL	Mid Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. MIDSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

32.3.8 Select 1 register (C55FMC_SEL1)

The Select 1 (SEL1) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHSEL	High Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5.

Table continues on the next page...

C55FMC_SEL1 field descriptions (continued)

Field	Description
	The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. HIGHSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected for erase 1 The block is selected for erase

32.3.9 Select 2 register (C55FMC_SEL2)

The Select 2 (SEL2) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL2 field descriptions

Field	Description
0–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

32.3.10 Select 3 register (C55FMC_SEL3)

The Select 3 (SEL3) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																A256KSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved, reset to 0.
16–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 1 The block is selected for erase. 0 The block is not selected.

32.3.11 Address register (C55FMC_ADR)

The Address register (ADR) provides the first failing address in the event of a failure (ECC or PGM/Erase state machine), as well as single bit correction information.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAD	aH	aM	aL	a256k	a64k	a32k	a16k	ADDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR													0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_ADR field descriptions

Field	Description
0 SAD	UTest NVM Address. The SAD bit qualifies the address captured during an ECC Event Error, Single Bit Correction, or State Machine operation. SAD has the same characteristics as ADDR related to capturing of addresses. The SAD field is not writable. 0 Address captured is from main array Space. 1 Address captured is from UTest NVM array space.
1 aH	Address High region. The aH bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aH is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from high address region. 1 Address captured or to be accessed is from high address region.
2 aM	Address Mid region. The aM bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aM is set, then the ADDR field maps to that region. See ADDR description for more information.

Table continues on the next page...

C55FMC_ADR field descriptions (continued)

Field	Description
	0 Address captured or to be accessed is not from mid address region. 1 Address captured or to be accessed is from mid address region.
3 aL	Address Low region. The aL bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aL is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from low address region. 1 Address captured or to be accessed is from low address region.
4 a256k	Address 256 KB region. The a256k bit qualifies the address field (ADDR) to the region. If a256k is set, then the ADDR field maps to that region. See ADDR description for more information. The a256k register is writable. 0 Address captured or to be accessed is not from 256 KB region. 1 Address captured or to be accessed is from 256 KB region.
5 a64k	Address 64 KB region. The a64k bit qualifies the address field (ADDR) to the region. If a64k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 64 KB region. 1 Address captured or to be accessed is from 64 KB region.
6 a32k	Address 32 KB region. The a32k bit qualifies the address field (ADDR) to the region. If a32k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 32 KB region. 1 Address captured or to be accessed is from 32 KB region.
7 a16k	Address 16 KB region. The a16k bit qualifies the address field (ADDR) to the region. If a16k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 16 KB region. 1 Address captured or to be accessed is from 16 KB region.
8–28 ADDR	Address. The ADR provides the first failing address in the event of ECC event error (EER set), Single Bit Correction (SBC set), as well as providing the address of a failure that may have occurred in a state machine operation (PEG cleared). ECC event errors take priority over Single Bit Corrections, which take priority over state machine errors. This is especially valuable in the event of an RWW operation, where the read senses an ECC error or Single Bit correction, and the state machine fails simultaneously. The failing address for ECC event error is held until EER is cleared. The failing address for Single Bit Correction is held until an ECC event error, or until SBC is cleared. State machine address is held until an ECC event error or Single Bit Correction event occurs, or until the next state machine event (PEG cleared) occurs. This address is always a doubleword address that selects 64 bits. ADDR[23:3] is an offset from a base address of 0x0 for each block size region. The aH, aM, aL, a16k, a32k, a64k, and a256k qualify the block size region the ADDR field.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

32.3.12 UTest 0 register (C55FMC_UT0)

The User Test 0 (UT0) register provides a means to control UTest. UTest mode gives the ability to perform test features on the flash memory. This register is only writable when the flash memory is put into UTest mode by writing a passcode.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UTE	SBCE	0											CPR	CPA	CPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						NAIBP	AIBPE	0	AISUS	MRE	MRV	0	AIS	AIE	AID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

C55FMC_UT0 field descriptions

Field	Description
0 UTE	U-Test Enable. This status bit gives indication when U-Test is enabled. All bits in UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. The UTE password will only be accepted if PGM = 0 and ERS = 0 (program and erase are not being requested). UTE can only be cleared if AID = 1, MRE and AIE = 0. While clearing UTE, writes to set AIE or MRE will be ignored. For UTE, the password 0xF9F9_9999 must be written to the UT0 register. 0 All bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are locked 1 Bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are unlocked
1 SBCE	Single Bit Correction Enable. SBCE enables Single Bit Correction results to be observed in SBC. ECC corrections that occur when SBCE is cleared will not be logged.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	0 Single Bit Corrections observation is disabled. 1 Single Bit Correction observation is enabled.
2–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CPR	Customer Programmable Read Voltage and Reference Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Read Voltage and Reference Error detection will be noted. This bit may not be set while CPE and CPA are set. If CPE or CPA are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled
14 CPA	Customer Programmable Address Encode Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Address Encode Error detection will be noted. This bit may not be set while CPE and CPR are set. If CPE or CPR are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a EDC after ECC Error detection will be noted. This bit may not be set while CPR and CPA are set. If CPR or CPA are high, this bit will be locked. If attempts are made o write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 NAIBP	Next Array Integrity Break Point. If AIBPE is set, NAIBP will be set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to 1, but may be cleared to 0. Clearing NAIBP to 0, will enable the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID will be set with NAIBP remaining 0. 1 Array Integrity state machine is at a break point. 0 Array Integrity state machine is not currently at a break point.
23 AIBPE	Array Integrity Break Point Enable. If you want to enable breakpoints during an array integrity test, AIBPE may be set. See NAIBP description for more information about array integrity breakpoints. 1 Array Integrity breakpoints are enabled during Array Integrity Checks. 0 Array Integrity breakpoints are not enabled.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 AISUS	Array Integrity Suspend. AISUS enables a suspend of an Array Integrity Operation. Array Integrity may be suspend by Setting AISUS, and resumed by clearing AISUS. AISUS is writeable only when AID is low. AISUS is clearable only when AID is high. Attempting to write AISUS and AIE on the same clock cycle will result in only AIE getting written.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	1 Array integrity sequence is suspended. 0 Array integrity sequence not suspended.
26 MRE	Margin Read Enable. MRE combined with MRV enables user margin reads to be done. Normal user reads are not affected by MRE, although user reads while the margin read operation is ongoing are not supported. MRE is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 Margin reads are enabled. 0 Margin reads are not enabled.
27 MRV	Margin Read Value. MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 One's margin reads are requested. 0 Zero's margin reads are requested.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AIS	Array Integrity Sequence. AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences that normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is logically sequential. It should be noted that the time to run a sequential sequence is shorter than the time to run the proprietary sequence. AIS is not writeable if AIE is high. 1 Array integrity sequence is sequential. 0 Array integrity sequence is proprietary sequence.
30 AIE	Array Integrity Enable. AIE set to one starts the array integrity check done on all selected blocks. The address sequence is determined by AIS, and the MISR (UM0 through UM9) can be checked after the operation is complete, to determine if a correct signature has been obtained. Once an array integrity operation is requested (AIE=1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE is not simultaneously writable to a 1 as UTE is being cleared to a 0. 0 Array integrity checks are not enabled. 1 Array integrity checks are enabled.
31 AID	Array Integrity Done. AID is cleared upon an array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set), an abort is requested or a suspend is requested. AID cannot be written, and is status only. 0 Array integrity check is ongoing. 1 Array integrity check is done.

32.3.13 UMISR register (C55FMC_UM*n*)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 58h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_UM*n* field descriptions

Field	Description
0–31 MISR	MISR[31:0]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

32.3.14 UMISR register (C55FMC_UM9)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															MISR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_UM9 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MISR	MISR[288]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained

32.3.15 Over-Program Protection 0 register (C55FMC_OPP0)

The Over-Program Protection 0 (OPP0) register provides a means to protect blocks from being over-programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWOPP														MIDOPP															
W																																
Reset	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- MIDOPP field: The reset value of the MIDOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.
- LOWOPP field: The reset value of the LOWOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWOPP	Low Block Over-Program Protection[13:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The LOWOPP register is not writable, and is status only. The default value of the LOWOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.
16–31 MIDOPP	Mid Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted.

Table continues on the next page...

C55FMC_OPP0 field descriptions (continued)

Field	Description
	The MIDOPP register is not writable, and is status only. The default value of the MIDOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

32.3.16 Over-Program Protection 1 register (C55FMC_OPP1)

The Over-Program Protection 1 (OPP1) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHOPP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- HIGHOPP field: The reset value of the HIGHOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHOPP	High Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the High Blocks starts at HIGHOPP[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The HIGHOPP register is not writable, and is status only. The default value of the HIGHOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

32.3.17 Over-Program Protection 2 register (C55FMC_OPP2)

The Over-Program Protection 2 (OPP2) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

> The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP2 field descriptions

Field	Description
0–31 A256KOPP	256K Block Over-Program Protection[31:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 0 The corresponding block is available to be over-programmed.

32.3.18 Over-Program Protection 3 register (C55FMC_OPP3)

The Over-Program Protection 3 (OPP3) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

> The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A256KOPP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KOPP	256K Block Over-Program Protection[47:32]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at 256KOPP[0] and continues until all blocks are accounted. The 256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

32.4 Functional Description

The embedded flash memory consists of address spaces in four groupings:

- Low address space
- Mid address space
- High address space
- 256 KB address space

The main address space is divided into partitions. Each address space mentioned above consists of a partition pair. Partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a write (program or erase) to a given partition, it can simultaneously perform a read from any other partition. For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written). For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The main address space is also divided into blocks to implement independent erase and program protection. The UTest NVM block also exists as a block, and has independent program protection. The UTest NVM block is included to support systems that require nonvolatile memory (NVM) for security or to store system initialization information.

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit-by-bit basis in Module Configuration Register (C55FMC_MCR). The write locks do not consider the effects of trying to write two or more bits simultaneously. The embedded flash memory does not allow bits to be written simultaneously which put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in the following table.

Table 32-1. MCR bit set/clear priority levels

Priority level	MCR Bit(s)
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If two or more MCR bits are written simultaneously only the bit with the lowest priority number is accepted for modification. Setting two bits with the same priority number is prevented by existing write locks or do not put the flash memory in an illegal state.

For example, setting ERS and PGM simultaneously results in only ERS being set. Attempting to clear EHV while setting PSUS results in EHV being cleared, while PSUS is unaffected.

Each read of the embedded flash memory retrieves a page, or eight consecutive words (256 bits), of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits [4:2]. The flash memory page read architecture easily supports both cache and burst mode at the BIU level for high-speed read applications.

The embedded flash memory supports fault tolerance through error correction code (ECC) and error detection. ECC implemented within the embedded flash memory corrects single-bit failures and detects double-bit failures.

A software mechanism is provided to independently lock and unlock each block.

Program and erase of the embedded flash memory requires multiple system clock cycles to complete. The program and erase sequence may be suspended or aborted.

The embedded flash memory may reside in various modes. The modes that are available include:

- User mode
- Low Power mode
- UTest mode

Each of these modes is discussed in detail in the following sections.

32.4.1 User Mode

In user mode the embedded flash memory may be read and written (register writes and interlock writes), programmed or erased. The following sub-sections define all actions that may be performed in user mode.

32.4.1.1 Read and write

The default state of the embedded flash memory is read. The main and UTest NVM address space can be read only in the read state. The MCR is always available for read, except when the embedded flash memory is in low power mode. The embedded flash memory enters the read state on reset. The embedded flash memory is in the read state under four sets of conditions:

- The read state is active when the embedded flash memory is enabled (User Mode Read).
- The read state is active when MCR[PGM] or MCR[ERS] are high and high-voltage operation is ongoing (read-while-write).

NOTE

Reads done to the partition(s) being operated on (either erased or programmed) result in an error and the MCR[RWE] bit is set.

- The read state is active when MCR[PGM] and MCR[PSUS] are high (program suspend).
- The read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (erase suspend).

In embedded flash memory, flash core reads return 256 bits (1 page). Register reads return 32 bits of data.

NOTE

> Flash core reads are done through the BIU. In many cases the BIU does read page buffering to allow sequential reads to be done with higher performance. This could provide a data coherency issue that must be handled with software. Data coherency may be an issue after a program or erase operation, as well as UTest NVM block operations.

In user mode, registers may be written. Array may be written to do interlock writes.

Register reads to unmapped register address space return all zeroes.

Register writes to unmapped register address space have no effect.

Interlock writes that are attempted during a high-voltage operation (MCR[EHV] = 1 or MCR[DONE] =0) will result in the interlock write being ignored, and address and data will not be updated.

NOTE

> Care must be taken when doing 32-bit interlock writes. Performing two 32-bit interlock writes to the same word location is not supported. In addition, mixing 32-bit interlock writes with 64-bit interlock writes to the same double-word location is not supported. If these combinations are done, and a program operation is executed, the result will be MCR[PEG] = 0. See the Module Configuration Register (MCR) section for more information.

32.4.1.2 Program

A flash memory program sequence operates on any page within the flash core. Within a page, up to eight words may be altered in a single program operation. Also, up to four pages can be altered in a single program operation. Whenever the array is programmed, the ECC bits also get programmed. ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed, since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be from 64 bits to 1024 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64-bit ECC segment between erases.

Warning

In rare cases over-programming of a 64-bit ECC segment may be done (EEPROM emulation).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

NOTE

If a logic 0 is attempted to be over-programmed by a logic 1, the resulting operation will fail (MCR[PEG] = 0), and the 0's that are interlocked will be merged (ORed) with 0's that are already present in the 64-bit ECC segment, unless the block is designated as an Over-Program Protected block.

Addresses in locked/disabled blocks cannot be programmed. Values may be programmed in any or all of eight words, within a page, with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. The program operation consists of the following sequence of events:

1. Change the value in the MCR[PGM] bit from a 0 to a 1.

   ### NOTE

   Ensure the block that contains the address to be programmed is unlocked.

2. Write the first address to be programmed with the program data. The embedded flash memory latches address bits [22:7] and SoC-specific UTest NVM enable at this time. The embedded flash memory latches data written as well. This write is referred to as a program data interlock write. An interlock write may be done as a 64-bit transaction or a 32-bit transaction. Refer to the Flash Memory Controller chapter for information on the size of writes that are allowed.
3. If more than one word or doubleword is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. The embedded flash memory ignores address bits [22:7] and SoC-specific UTest NVM enable for program data writes. All unwritten data words default to 0xFFFF_FFFF.
4. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the MCR[PGM] bit to terminate the program sequence.

The first write after a program is initiated determines the quad-page address to be programmed. Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. The interlock write determines if the UTest NVM or normal array space is to be programmed by sampling SoC-specific UTest NVM enable and causing MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the values in MCR[PEAS] may be modified via the program interlock write, enabling erase-suspended programs to and from UTest NVM space.

An interlock write must be performed before setting MCR[EHV]. A program sequence may be terminated by clearing MCR[PGM] prior to setting MCR[EHV].

After the interlock write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024-bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are done to the same location, the data for the last write is used in programming.

While DONE is low, EHV is high, and PSUS is low, then EHV may be cleared, resulting in a program abort. A program abort forces the embedded flash memory to step 8 of the program sequence. An aborted program results in PEG being set low, indicating a failed operation. The data space being operated on before the abort contains indeterminate data. A program sequence may not be aborted while in program suspend.

Warning

Aborting a program operation leaves the flash core addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

32.4.1.2.1 Program software locking

A software mechanism is provided to independently lock/unlock blocks against program and erase.

Software locking is done through the LOCK0, LOCK1, LOCK2, and LOCK3 registers. These may be written through register writes, and may be read through register reads.

32.4.1.2.2 Program hardware locking

Hardware locking which only affects the program operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

32.4.1.2.3 Program suspend/resume

The program sequence may be suspended to allow read access to the flash core. It is not possible to erase during a program suspend, or to program during a program suspend. These operations are prevented by register locking described in the Module Configuration Register (MCR) section. Read-while-write operation may also be used to read the array during a program sequence, providing the read is to a different partition.

A program suspend can be initiated by changing the value of the MCR[PSUS] bit from a 0 to a 1. MCR[PSUS] can be set high at any time when MCR[PGM] and MCR[EHV] are high. A 0 to 1 transition of MCR[PSUS] causes the embedded flash memory to start the sequence to enter program suspend, which is a read state. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended. At this time flash core reads may be attempted. Once suspended, the flash core may only be read. During suspend, all reads to flash core locations targeted for program, and blocks targeted for erase, return indeterminate data.

The program sequence is resumed by writing a logic 0 to MCR[PSUS]. MCR[EHV] must be set to a 1 before clearing MCR[PSUS] to resume operation. When the operation resumes, the embedded flash memory continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

NOTE

Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory will respond by completing the operation with a fail code (MCR[PEG] = 0). Although suspend frequency in not limited, enabling at least 20uS between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation.

32.4.1.2.4 Over-program protection enable

One-time programming can be enabled on a block basis. In an over-program protection enabled block, any doubleword which has already been programmed cannot be programmed again. The over-program protection enable does not affect erase operation. Attempts to over-program will result in MCR[PEG] being cleared. If any doubleword within the quad-page has an over-program protection violation, MCR[PEG] will be cleared, and no doublewords will be programmed.

One-time programming can be enabled for 16 KB, 32 KB, 64 KB, or 256 KB blocks.

Over-program protection is enabled by sideband signals, and if enabled, the program operation will be modified to check for programmed bits prior to doing a high-voltage program event.

32.4.1.2.5 Successful program address

To enable the SoC to create logic to authenticate operations with a program (such as in a diary operation), the flash module provides the last successful address programmed as a sideband signal. This value will be held until the next successful program. A successful program is defined as a program operation of data that is not all 1's to an unlocked block, and the operation results in MCR[PEG] returning a 1. If any doubleword within the pages being programmed is interlocked with all 1's, the sideband signal will not be updated. An address of all 1's, and each of the block size selects selected, is the default (and invalid) address after reset.

32.4.1.3 Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the main address space. The erase sequence is fully automated within the flash memory. Blocks to be erased must be selected prior to initiating the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest (low first, mid second, high last, 16 KB first, 32 KB second, 64 KB third, last blocks are 256 KB). The erase sequence consists of the following events:

1. Change the value in the MCR[ERS] bit from 0 to 1.
2. Select the block or blocks to be erased by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers.

NOTE

Lock and Select are independent. If a block is selected and locked, no erase occurs.

3. Write to any address in flash memory. This is referred to as an erase interlock write. An erase interlock write to UTest NVM space is not allowed.
4. Write a logic 1 to the MCR[EHV] bit to start an internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR[ERS] bit to terminate the erase.

After setting ERS, one write, referred to as an interlock write, must be performed before EHV can be set to a 1. Data words written during erase sequence interlock writes are ignored. The erase sequence may be terminated by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV, assuming DONE is low, EHV is high, and ESUS is low. An erase abort forces the embedded flash memory to step eight of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. An erase sequence may not be aborted while in erase suspend.

32.4.1.3.1 Erase software locking

Software Locking affect erase operation. For details on this see Program software locking.

32.4.1.3.2 Erase hardware locking

Hardware locking which only affects the erase operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

32.4.1.3.3 Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core. The erase sequence may also be suspended to program (Erase-Suspended Program) the flash core. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to flash core locations targeted for program and blocks targeted for erase return indeterminate data.

Read-while-write operation may also be used to read the array during an erase sequence, provided the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the embedded flash memory to start the sequence which places it in erase suspend. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended and further actions are attempted. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence, MCR[EHV] must first be cleared. If a program sequence is initiated, the values of SoC-specific UTest NVM enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its erase value. Flash core reads that occur while MCR[ESUS] = 1 from the block(s) being erased will return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase-suspended program) before MCR[ESUS] can be cleared to resume the operation. The embedded flash memory continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

Warning

In an erase-suspended program, programming flash core locations in blocks which were being operated on will respond by completing the operation with a fail code (MCR[PEG] = 0). Programming voltages will not be applied to the memory array in this case.

Warning

Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory will respond by completing the operation with a fail code (MCR[PEG] = 0). Although suspend frequency is not limited, enabling at least 5mS between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation.

32.4.2 Low Power mode

In Low Power mode, the embedded flash memory enables VDDF power gating, and disables circuits on the VFLASH domain. No reads from or writes to the embedded flash memory are possible when in Low Power mode.

Prior to entering Low Power Mode, it is required that all program, erase, and UTest operations be stopped prior to entering the mode. If Low Power mode is entered while an operation is suspended, entering Low Power mode will have the same effect as a reset during suspend.

When in Low Power mode, register access is prevented. Flash core accesses are also prevented until power mode is exited. Flash core reads and writes may occur after power mode is exited.

The embedded flash memory returns to a post-reset state in all cases.

32.4.3 UTest mode

UTest mode is a mode into which customers can put the embedded flash memory so as to do specific tests that check the integrity of the embedded flash memory.

32.4.3.1 Array integrity self check

Array integrity is checked using a predefined address sequence (based on UT0[AIS]), and this operation is executed on selected blocks. The data to be read is customer-specific - user code may be programmed into the flash memory and the correct MISR signature is calculated based on that code. Any random or nonrandom code is valid. Once the operation is completed, the results of the reads can be checked by reading the MISR value, to determine if an incorrect read or ECC detection was noted. Array integrity MISR value is calculated after ECC detection and correction. Array integrity requires that the Read Wait States and Address Pipelined control registers in the BIU be set to match the system frequency being used. The array integrity check consists of the following sequence of events:

1. Enable UTest mode.

2. Select the block or blocks to be receive the array integrity check by writing 1's to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers. Blocks selected for array integrity check do not need to be unlocked.

   NOTE

   Unselected blocks are still read as part of the array integrity sequence, to enable full transition coverage. The result read on unselected blocks will not be captured in the MISR. Selecting fewer blocks may not result in a faster array integrity execution time, depending on the combinations of blocks selected and the sequence.

   NOTE

   It is not possible to do array integrity operations on the UTest NVM block.

3. If desired, set the UT0[AIS] bit to 1 for sequential addressing only.

   NOTE

   For normal integrity checks of the flash memory, sequential addressing is recommended. If it is required to more fully check the read path (in a diagnostic mode), it is recommended that AIS be left at 0, to use the address sequence that checks the read path more fully, and examine read transitions. This sequence takes more time.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, ensure that UT0[SBCE] is set.
6. Set the UT0[AIE] bit.
   a. If desired, the array integrity operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and then continuing to the next step. It should be noted that in the event of an aborted array integrity check the MISR registers will contain a signature for the portion of the operation that was completed prior to the abort, and will not be deterministic. Prior to doing another array integrity operation, the UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8, and UM9 registers may need to be initialized to the desired seed value by doing register writes.
   b. If desired, the array integrity operation may be suspended prior to UT0[AID] going high. UT0[AISUS] may be set to request an array integrity suspend. After the UT0[AISUS] bit is set, the user should wait for UT0[AID] bit to go high, which indicates the flash has entered the suspend state, and normal reads to the

flash may be done. Once [AID] goes high, UT0[AISUS] may be cleared to resume the array integrity sequence.

NOTE

User mode array reads requested during the array integrity test will be ignored, to ensure that the array integrity operation is not corrupted. The memory array will not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

7. Wait until the UT0[AID] bit goes high.
8. If breakpoints were enabled, check UT0[NAIBP], and if this is set to 1, the MCR[EER], MCR[SBC], and ADDR registers may be checked to determine the cause of the break and the address of the break. Prior to resuming the operation, clear MCR[SBC] or MCR[EER] bits. Then the operation may be resumed by clearing UT0[NAIBP]. Continue to wait until the UT0[AID] bit goes high. If breakpoints were not enabled, or if UT0[NAIBP] is low when UT0[AID] goes high, then the operation is complete. Continue to the next step.
9. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Array integrity reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

10. Write a logic 0 to the UT0[AIE] bit.

32.4.3.2 User margin read

User margin read may be done using the array integrity interface, and has all the associated features of the array integrity interface (MISR and Breakpoints). User margin reads are done at a read margin level checking for erased bits or programmed bits encroaching on the nominal read level. User margin read is a self-timed event, and is independent of system clocks or wait states selected. Margin ECC corrections and detections are noted during the user margin read test. Margin read MISR value is calculated after ECC detection and correction.

The data to be read is customer-specific, and user code may be programmed into the flash memory. Any random or non-random code is valid. Once the operation is completed, the margin read results can be checked by reading the MCR[EER] bit and the MCR[SBC] bits to determine if zero, one, or two bits are being detected by the margin read, as well as checking the MISR.

The use model for margin read is in the event of a user-detected single bit correction (through user reads). A margin read may be done to check for a possible second bit falling within the selected margin levels.

The procedure for doing margin reads is:

1. Enable UTest mode.
2. Select the block or blocks to receive margin read check by writing 1's to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers. Blocks selected for margin read do not need to be unlocked.

NOTE

Unselected blocks are still read as part of the margin read sequence, to enable full transition coverage. The result read on unselected blocks will not be captured in the MISR. Selecting fewer blocks will not result in a faster margin read execution time.

NOTE

It is not possible to do margin read operations on the UTest NVM block.

3. Set the UT0[AIS] bit to 1 for sequential addressing only.

NOTE

For margin read checks of the flash memory, sequential addressing is recommended. Setting AIS to 0 is possible for Margin Reads, but using the sequence takes more time, and is not recommended.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. Ensure that the MCR[EER] and MCR[SBC] bits are cleared.
6. If you want to detect single bits during margin read, set UT0[SBCE] to 1.
7. Set the UTO[MRE] bit.
8. Set the UT0[MRV] bit to the desired value depending on if it is desired to do one's margin or zero's margin.
9. Set the UT0[AIE] bit.

NOTE

User mode array reads requested during the margin read test will be ignored, to ensure that the margin read operation is not corrupted. The memory array will not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

NOTE

During Margin Read operations, with AID low, it is not recommended to attempt write operations to the MCR[SBC] and MCR[EER] bits. It is recommended that these bits only be written during suspend, breakpoints, or at completion of the Margin Read operation.

10. Wait until the UT0[AID] bit goes high.
    a. If breakpoints were enabled or a suspend was requested during margin read, the operation may be at a breakpoint or a suspend state. See Array integrity self check for more information.
11. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Margin reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

12. Write a logic 0 to the UT0[AIE] bit.

32.5 Initialization information

A reset is the highest priority operation for the embedded flash memory and terminates all other operations.

The embedded flash memory uses reset to initialize register and status bits to their default reset values. If the embedded flash memory is executing a program or erase operation (PGM or ERS = 1) and a reset is issued, the operation is aborted and the embedded flash memory disables the high-voltage logic without damage to the high-voltage circuits. Reset aborts all operations and forces the embedded flash memory into user mode ready to receive accesses.

After reset is requested, MCR[DONE] goes low, and remains low during reset and reset recovery. At the end of reset recovery, MCR[DONE] transitions from a 0 to a 1.

After reset is completed, register reads may be done, although it should be noted that registers that require updating from UTest NVM information, or other inputs, may not read updated values until MCR[DONE] transitions high.

During reset recovery, register writes are not allowed until the MCR[DONE] bit transitions high to indicate reset recovery is completed.

Warning

Resetting during a program or erase operation leaves the flash core blocks being programmed or erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

Chapter 33
Flash OTP Control

33.1 Introduction

The chapter describes access permissions for flash memory blocks.

33.2 One Time Programmable (OTP) Blocks

The flash memory is divided into several blocks, which can be used as one time programmable (OTP) blocks. If a block is designated to be OTP, it cannot be erased, and already written locations in the flash memory cannot be overwritten (overwrite protection). This applies even if the overwriting would solely clear bits that were still 1.

DCF records control which blocks are set to be OTP. Note that DCF records are applied only during device reset, so appending a record only takes affect after the next reset.

33.3 OTP DCF Records

33.3.1 OTP Record - Low and Mid Blocks

Each flash memory block can be set to be OTP (one time programmable). This record controls the OTP functionality of the low and middle blocks.

0	1	2:15	16:31
rsrvd	*rsrvd*	LOWBLOCKS	MIDBLOCKS

0:31
0100_0010h

Figure 33-1. OTP Record - Low and Mid Blocks

Programming a '1' via a DCF record means the block will be OTP and erase-locked.

33.3.2 OTP Record - High Blocks

Each flash memory block can be set to be OTP (one time programmable). This record controls the OTP functionality of the high blocks.

0:15	16:31
rsrvd	HIGHBLOCK

0:31
0100_0014h

Figure 33-2. OTP Record - High Blocks

The initial value of each OTP control bit is '0'. Programming a '1' via DCF record means the block will be OTP.

33.3.3 OTP Record - Lower 256k Blocks

Each flash memory block can be set to be OTP (one time programmable). This record controls the OTP functionality of the lower 256k blocks.

0:31
256BLCK_L

0:31
0100_0018h

Figure 33-3. OTP Record - Lower 256k Blocks

The initial value of each OTP control bit is '0'. Programming a '1' via DCF record means the block will be OTP.

Chapter 34
Decorated Storage Memory Controller (DSMC)

34.1 Introduction

This section details the hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called "*decorated storage*". It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A "decoration" is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l[b,h,w]d{cb}x rT,rB,rA` where the data size specifier is defined as 8-bit (`b = `byte), 16-bit (`h = `halfword) or 32-bit (`w = `word), and the three register specifiers include `rT` as the destination target register, `rB` as the effective address and `rA` as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st[b,h,w]d{cb}x rS,rB,rA` where the same data size specifiers are used, and the three register specifiers are `rS` as the source data register, `rB` as the effective address and `rA` as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the rB register (hence the `"x"` mnemonic suffix, signaling an "indexed" addressing mode) and the decoration is specified by the rA register. The core transmits the contents of the rA register as the decoration value along with the access address (register rB) to the DSMC.

The decorated load and store instruction formats are shown in the following figure.

Table 34-1. Power Architecture decorated load and store instruction formats

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1	rT					rA					rB					1	0	0	0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1	rT					rA					rB					1	0	0	0	1	0	0	0	1	1	/
lwdx	0	1	1	1	1	1	rT					rA					rB					1	0	0	1	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1	rT					rA					rB					1	0	0	0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1	rT					rA					rB					1	0	0	0	1	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1	rT					rA					rB					1	0	0	1	0	0	0	0	1	0	/
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
stbdx	0	1	1	1	1	1	**r**S					**r**A					**r**B					1	0	1	0	0	0	0	0	1	1	/
sthdx	0	1	1	1	1	1	**r**S					**r**A					**r**B					1	0	1	0	1	0	0	0	1	1	/
stwdx	0	1	1	1	1	1	**r**S					**r**A					**r**B					1	0	1	1	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1	**r**S					**r**A					**r**B					1	0	1	0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1	**r**S					**r**A					**r**B					1	0	1	0	1	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1	**r**S					**r**A					**r**B					1	0	1	1	0	0	0	0	1	0	/

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit "decoration". The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and/or operand data for the command.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (`l[b,h,w]dcbx`) and store (`st[b,h,w]dcbx`) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

34.2 DSMC block diagram

The following block diagram, Figure 34-1, shows the basic datapaths for the ECC logic in the DSMC. On both the read data path (hrdata) and the write data path (hwdata) there is logic to detect an ECC error and correct the data. The circle with X inside represents an XOR operation. The ecc_hmatrix and ecc_detect blocks are coefficient matrixes based on the Hsiao ECC algorithm. Notice in the recalculation of the s_hwchkbit[7:0] that there is a read chkbit from the slave (hrchkbit), a zero padded result contribution, and a contribution from the hwchkbit from the master. The details of the DSMC bit field and other operations are not shown (represented by the DSMC Operation box).

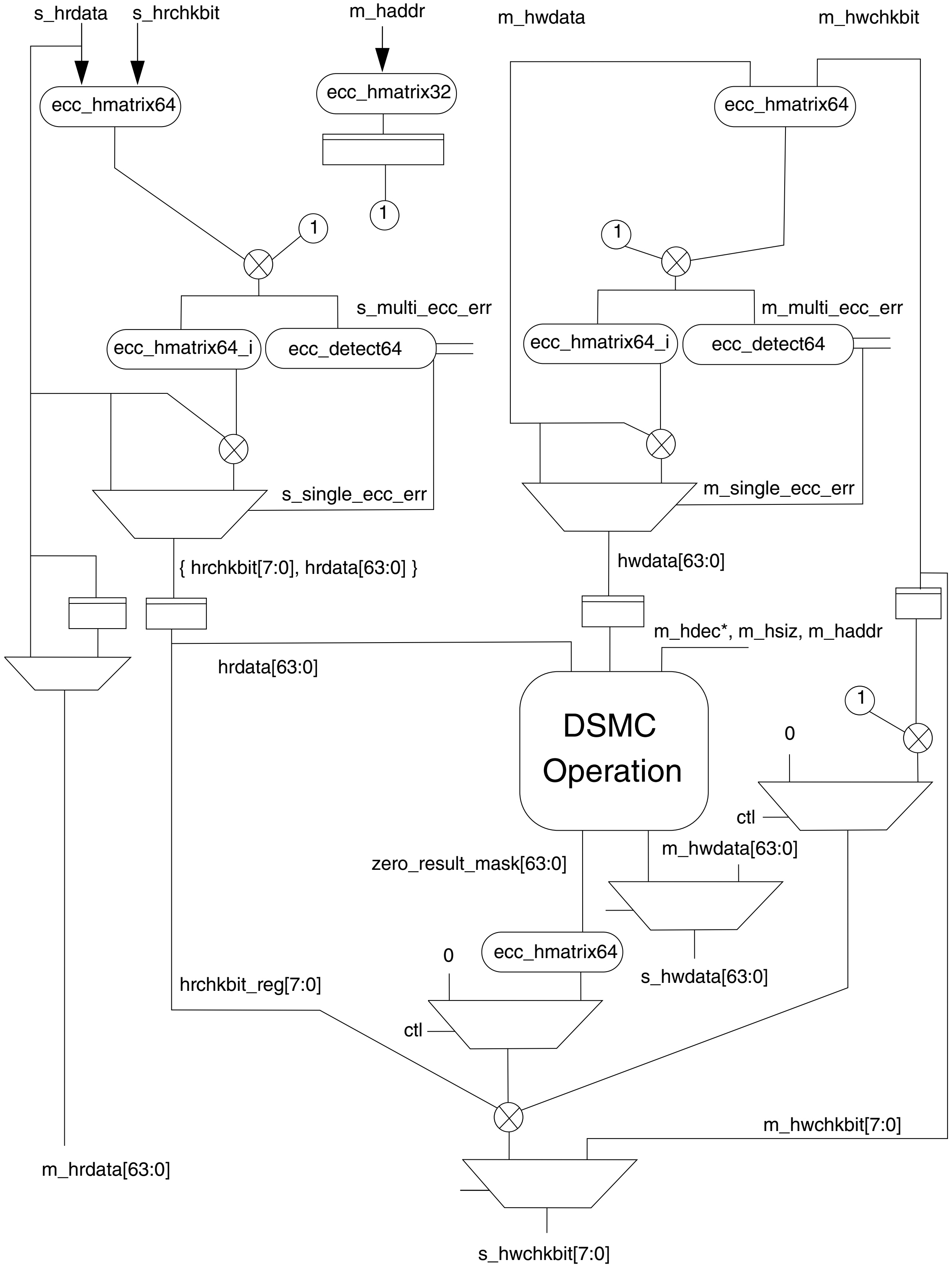


Figure 34-1. DSMC Block diagram

34.3 Decorated stores: `st[b,h,w]d{cb}x rS,rA,rB`

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a `<command>.<size>` syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
 8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
 8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- wdata = rS

34.3.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit memory container

Table 34-2. Decoration format: BFINS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bfins.b	0	0	0	0	0	0	0	SRTBIT			0	0	0	BFW			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.h	0	0	0	0	0	0	SRTBIT				0	0	BFW				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.w	0	0	0	0	0	SRTBIT					0	BFW					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command inserts a bit field defined by SRTBIT and BFW into the memory "container" defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read
if bfw == 0
         then bfw = container

if ((srtbit + bfw) <= container) // generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp  & ~mask// modify
    | wdata &  mask

mem[accessAddress, size] = tmp // memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd_efgh". For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz-_----,
   then destination is "xyzd_efgh"
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
   then destination is "axyz_efgh"
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
   then destination is "abxy_zfgh"
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
   then destination is "abcx_yzgh"
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
   then destination is "abcd_xyzh"
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_-xyz,
   then destination is "abcd_exyz"
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
   then destination is "abcd_efxy"
```

```
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
   then destination is "abcd_efgx"
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

34.3.1.1 Compare-and-Store (CAST)

Table 34-3. Decoration format: CAST

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cast.b	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CD8							
cast.h	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	CD16															
cast.w	1	0	0	1	CD28																											

This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size]  // memory read

if (size == 8)             // define compare_operand
    compare_operand = CD8
else if (size == 16)
          compare_operand = CD16
     else compare_operand = {0x0, CD28}// zero-filled data

if (tmp == compare_operand) // compare - "modify"
     mem[accessAddress, size] = wdata// memory write
else mem[accessAddress, size] = tmp
```

Note for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit compare operand.

34.3.1.1.1 Logical AND (AND)

Table 34-4. Decoration format: AND

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
and. {b,h,w}	1	0	1	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata      // modify
mem[accessAddress, size] = tmp// memory write
```

34.3.1.1.2 Logical OR (OR)

Table 34-5. Decoration format: OR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
or.{b,h,w}	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata         // modify
mem[accessAddress, size] = tmp// memory write
```

34.3.1.1.3 Logical Exclusive-OR (XOR)

Table 34-6. Decoration format: XOR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
xor.{b,h,w}	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata         // modify
mem[accessAddress, size] = tmp// memory write
```

34.3.1.2 Decorated Loads: `l[b,h,w]d{cb}x rT,rA,rB`

The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an "extract") operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

34.3.1.2.1 Simple Load (SLD)

Table 34-7. Decoration format: SLD

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sld. {b,h,w}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

34.3.1.2.2 Registers-and-memory exchange (SWAP)

Table 34-8. Decoration format: SWAP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
swap.b	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	WD8							
swap.h	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	**WD16**															
swap.w	0	1	0	1	WD28																											

This command loads the contents of the referenced memory location into the core's rT register and stores the source operand from the core's rA register (the decoration) into the memory location.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp

if (size == 8)             // store_operand - "modify"
    store_operand = WD8
else if (size == 16)
           store_operand = WD16
    else store_operand = {0x0, WD28} // zero-filled write data

mem[accessAddress, size] = store_operand// memory write
```

Note for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit store operand.

34.3.1.2.3 Load-and-Set-1(Bit) (LAS1)

Table 34-9. Decoration format: LAS1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	0	BIT			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp // rT = rdata = tmp
mask  = 1 << (container - bit - 1) // generate bit mask
tmp   = tmp | mask// modify
mem[accessAddress, size] = tmp// memory write
```

34.4 DSMC timing diagram

The following timing diagram, shown below, shows a decorated load followed by a decorated store instruction which is then followed by a normal load instruction. Both decorated instructions are read-modify-write instructions and thus they take at least two cycles to execute. In the first cycle an ECC calculation is performed on the read data as well as the write data. Then in the second cycle of the decorated instruction the decorated operation is performed and if it is a decorated store, the result is merged with the read data and sent to the write data bus along with the recalculated ECC write chkbits.

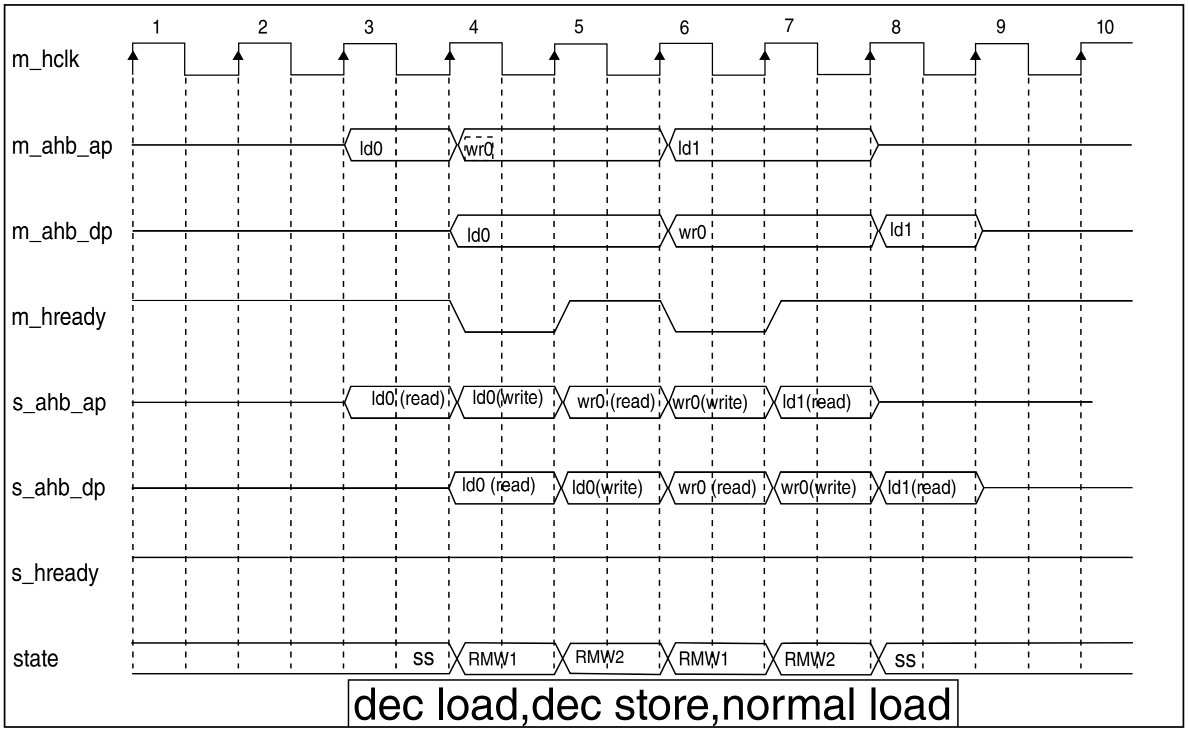


Figure 34-2. DSMC timing diagram

34.5 DSMC instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated memory controller only operates on aligned, single data transfers (misalignment and/or bursts are not supported and error terminated if attempted).

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

Chapter 35
ADC Configuration

35.1 Overview

There are four Analog to Digital Converter (ADC) modules located on the MPC5744P microcontroller. Each ADC consists of 16 channels. The particular ADC set has been chosen with 12-bit Successive Approximation Register (SAR) architecture to allow for precision conversion and sampling in less than 1 µs compounded.

To allow temperature readings while running your application, ADC0 has one channel for temperature sensor TSENS0, and ADC1 has one channel for temperature sensor TSENS1. In addition, all four ADCs have watchdog functionality that compares ADC results against predefined levels before results are stored in the appropriate ADC result location.

The ADCs operate on two different modes: Regular mode and Motor Control mode.

Regular mode features:

- Register based interface with the CPU: one result register per channel
- ADC state machine managing three request flows: regular command, hardware injected command, software injected command
- Selectable priority between software and hardware injected commands
- 16 analog watchdogs comparing ADC results against predefined levels (low, high, range)
- DMA compatible interface

Motor Control mode features:

- Triggered mode only
- Four independent result queues (1x16 entries, 2x8 entries, 1x4 entries)
- Result alignment circuitry (left justified, right justified)
- 32-bit read mode that allows channel ID on one of the 16-bit parts
- DMA compatible interfaces

Integrated with the four ADCs are two Cross Triggering Units (CTUs). Each CTU allows automatic generation of ADC conversion requests on user-selected conditions with minimal CPU intervention. These conversion requests, which occur in what is called CTU Control mode, take place during the PWM period for dynamic configuration.

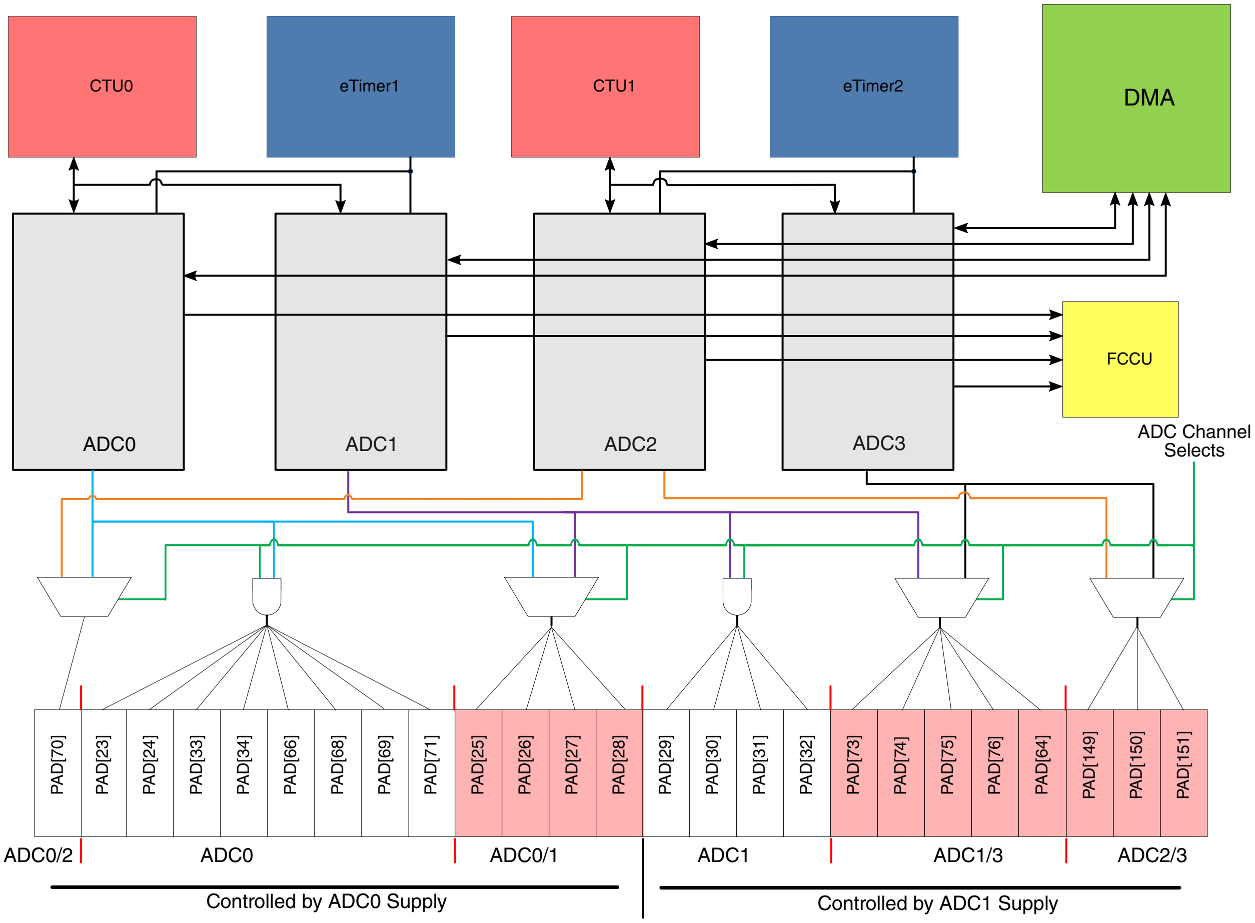
35.2 ADC configuration

35.2.1 Block diagram

The diagram below shows the four different ADCs and their interaction with the other modules on the device. The appropriate pin muxing is also highlighted in the diagram and is described in ADC pin muxing.

NOTE

All four ADCs have only two possible ADC supplies: ADC0 and ADC1. These two supplies must be enabled to use the ADC functionality.



35.2.2 CTU interface

Each ADC is controlled by the CTU (in CTU Control Mode). In this mode, the CTU can control each ADC by sending an ADC command. During the CTU Control Mode, the CPU is able to write in the ADC registers but it can not start a new conversion. For the MPC5744P device the CTU0 is the controller for ADC0 and ADC1 whilst the CTU1 controls the ADC2 and ADC3 modules.

NOTE

CTU Clock (MC_CLK) and ADC_CLK should either be same and synchronous or CTU can also operate with ADC_CLK being an integer plus half (1.5, 2.5,3.5, ...) of MC_CLK clock.

35.2.3 eTimer interface

All four ADCs allow an external trigger source to start a conversion. This feature is enabled through the injection external trigger enable (JTRGEN) bit in the Main Configuration Register (MCR) of the ADC. For the MPC5744P this external trigger feature is connected to timer channel 5 of an eTimer module, through which the eTimer sends an external trigger event. The eTimer1 is connected to ADC0 and ADC1, and the eTimer2 is connected to the ADC2 and ADC3 modules.

35.2.4 DMA interface

For each ADC a DMA request can be performed after the conversion of every channel.

NOTE

For 1N65H, the ADC_DMAE[DCLR] bit is RESERVED and should not be programmed by software.

35.2.5 FCCU interface

Each ADC on the MPC5744P device is connected to the Fault Collection and Control Unit (FCCU) and has the ability to create both critical and non-critical faults. For more on what these faults mean, please refer to the FCCU chapter.

35.3 ADC pin muxing

The following table provides ADC pin multiplexing details.

Table 35-1. ADC pin muxing

ADC		INT/EXT	Signal source		Package Availability
ADC Instance	Channel No		Source	Ext Pin	
Shared external channels - ADC0/1					
ADC0	11	EXT	ADC0_ADC1_AN[11]	PAD[25]	144LQFP 257BGA
ADC1	11				
ADC0	12	EXT	ADC0_ADC1_AN[12]	PAD[26]	144LQFP 257BGA
ADC1	12				
ADC0	13	EXT	ADC0_ADC1_AN[13]	PAD[27]	144LQFP

Table continues on the next page...

Table 35-1. ADC pin muxing (continued)

ADC		INT/EXT	Signal source		Package Availability
ADC Instance	Channel No		Source	Ext Pin	
ADC1	13				257BGA
ADC0	14	EXT	ADC0_ADC1_AN[14]	PAD[28]	144LQFP
ADC1	14				257BGA
Shared external channels - ADC0/2					
ADC0	4	EXT	ADC0_ADC2_AN[4]	PAD[70]	144LQFP
ADC2	4				257BGA
ADC0 External Channels					
ADC0	0	EXT	ADC0_AN[0]	PAD[23]	144LQFP 257BGA
ADC0	1	EXT	ADC0_AN[1]	PAD[24]	144LQFP 257BGA
ADC0	2	EXT	ADC0_AN[2]	PAD[33]	144LQFP 257BGA
ADC0	3	EXT	ADC0_AN[3]	PAD[34]	144LQFP 257BGA
ADC0	5	EXT	ADC0_AN[5]	PAD[66]	144LQFP 257BGA
ADC0	6	EXT	ADC0_AN[6]	PAD[71]	144LQFP 257BGA
ADC0	7	EXT	ADC0_AN[7]	PAD[68]	144LQFP 257BGA
ADC0	8	EXT	ADC0_AN[8]	PAD[69]	144LQFP 257BGA
ADC0	9	EXT	reserved	--	
ADC0	10	INT	Bandgap Reference PMC	--	
ADC0	15	INT	TSENS0	--	
ADC1 External Channels					
ADC1	0	EXT	ADC1_AN[0]	PAD[29]	144LQFP 257BGA
ADC1	1	EXT	ADC1_AN[1]	PAD[30]	144LQFP 257BGA
ADC1	2	EXT	ADC1_AN[2]	PAD[31]	144LQFP 257BGA
ADC1	3	EXT	ADC1_AN[3]	PAD[32]	144LQFP 257BGA
ADC1	9	EXT	reserved	--	
ADC1	10	INT	Bandgap Reference PMC	--	

Table continues on the next page...

Table 35-1. ADC pin muxing (continued)

ADC		INT/EXT	Signal source		Package Availability
ADC Instance	Channel No		Source	Ext Pin	
ADC1	15	INT	TSENS1	--	
Shared external channels - ADC1/3					
ADC1	4	EXT	ADC1_ADC3_AN[4]	PAD[75]	144LQFP 257BGA
ADC3	3				
ADC1	5	EXT	ADC1_ADC3_AN[5]	PAD[64]	144LQFP 257BGA
ADC3	4				
ADC1	6	EXT	ADC1_ADC3_AN6]	PAD[76]	144LQFP 257BGA
ADC3	5				
ADC1	7	EXT	ADC1_ADC3_AN[7]	PAD[73]	144LQFP 257BGA
ADC3	6				
ADC1	8	EXT	ADC1_ADC3_AN[8]	PAD[74]	144LQFP 257BGA
ADC3	7				
Shared external channels - ADC2/3					
ADC2	0	EXT	ADC2_ADC3_AN[0]	PAD[149]	257BGA
ADC3	0				
ADC2	1	EXT	ADC2_ADC3_AN[1]	PAD[150]	257BGA
ADC3	1				
ADC2	2	EXT	ADC2_ADC3_AN[2]	PAD[151]	257BGA
ADC3	2				
ADC2 External Channels					
ADC2	3	EXT	Reserved for factory test only	N/A	
ADC2	10	INT	Bandgap reference PMC	N/A	
ADC2	14	EXT	Reserved for factory test only	N/A	
ADC3 External Channels					
ADC3	10	INT	Bandgap Reference PMC	N/A	
ADC3	11	EXT	Reserved for factory test only	N/A	
ADC3	12	EXT	Reserved for factory test only	N/A	
ADC3	13	EXT	Reserved for factory test only	N/A	
ADC3	14	EXT	Reserved for factory test only	N/A	
ADC0 and ADC1 Presampling					
ADC0	PRESx	INT	VREFP_ADC0	N/A	
ADC0	PRESx	INT	VREFN_ADC0	N/A	
ADC1	PRESx	INT	VREFP_ADC1	N/A	
ADC1	PRESx	INT	VREFN_ADC1	N/A	
ADC2 and ADC3 Presampling					
ADC2	PRESx	INT	VREFP_ADC2	N/A	
ADC2	PRESx	INT	VREFN_ADC2	N/A	

Table continues on the next page...

Table 35-1. ADC pin muxing (continued)

ADC		INT/EXT	Signal source		Package Availability
ADC Instance	Channel No		Source	Ext Pin	
ADC3	PRESx	INT	VREFP_ADC3	N/A	
ADC3	PRESx	INT	VREFN_ADC3	N/A	

35.3.1 ADC channel conversion

For ADC channel conversion:

- Both Vref VDD_HV_ADRE0 and VDD_HV_ADRE1 must be powered up.
- Both must be at the same level

Chapter 36
Analog-to-Digital Converter (ADC)

36.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The analog-to-digital converter (ADC) uses Successive Approximation Register (SAR) architecture and provides accurate and fast conversions for a wide range of applications.

36.2 Overview

36.2.1 Parts of the ADC

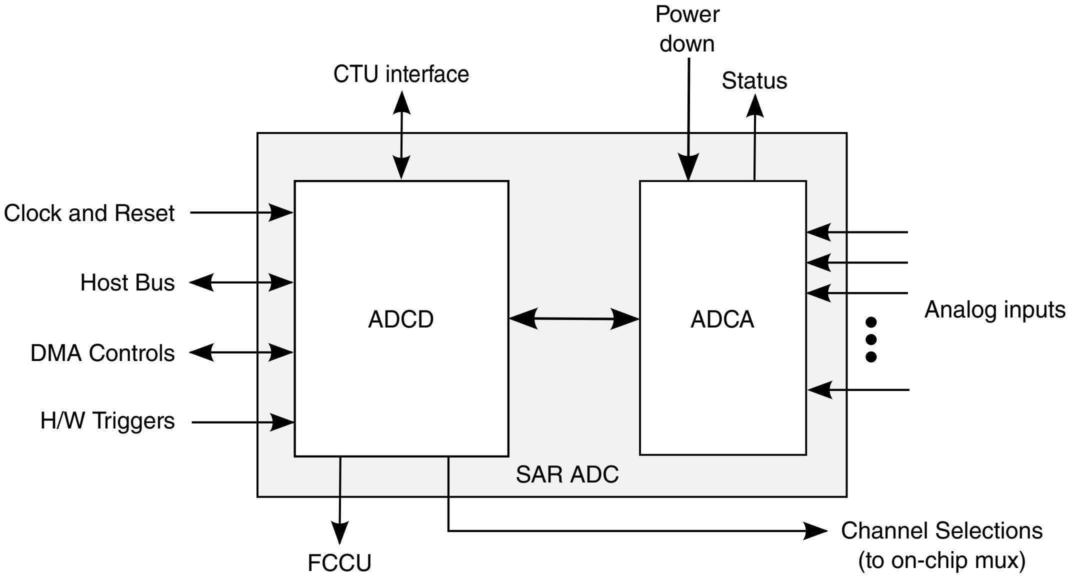


Figure 36-1. ADC high-level block diagram

The ADC is comprised of:

- An ADC digital interface (ADCD). The ADCD holds the configuration, control, and status registers accessible to software via the host bus. The converted value is stored in a register and then transferred through the DMA or host access.
- An ADC analog hard macrocell (ADCA). The analog channel inputs are fed to the inputs of the ADCA. Each channel is sampled for a specific duration and compared with an analog voltage generated with digital code via a digital-to-analog converter (DAC) in the ADCA. The comparison result is given to the ADCD to generate the converted digital value.

36.2.2 ADC clock signals

The ADC uses the following clock signals:

- Bus clock—bus clock at ADCD. Used to access the registers and the operating clock of inter-module interfaces.
- AD_clk—derived from *bus clock*, this signal serves as the operating clock for the ADCA and the SAR controller inside the ADCD.

The relation between the two clocks is determined by the setting of the ADC_MCR[ADCLKSEL] field.

- If ADC_MCR[ADCLKSEL] is 1, the clock signals are the same
- If ADC_MCR[ADCLKSEL] is 0, AD_clk is half of the bus clock.

NOTE

1. Regardless of the setting of the ADC_MCR[ADCLKSEL] field, the maximum clock speed for the ADC clock (AD_clk) is 80 MHz.
2. Depending on your chip's configuration and timing needs, MCR[ADCLKSEL] may always need to be 1.

If the bus clock for the microcontroller is 80 MHz and ADC_MCR[ADCLKSEL] is 1, the ADC clock (AD_clk) is 80 MHz.

If the bus clock for the microcontroller is 80 MHz and ADC_MCR[ADCLKSEL] is 0, the ADC clock (AD_clk) is 40 MHz.

See ADCLKSEL for details on the ADC_MCR register fields.

36.2.3 About ADC channels

36.2.3.1 Channel configuration

ADC channels can be configured in the following ways:

- The sampling duration is controlled independently, through programming the Conversion Timing Register (CTR).
- Separate mask registers can be programmed to configure the channels to be converted (each bit in these registers represents one channel).
- Temperature Sensor (TSENS) and other special channels (band gap, supply etc.) are available on specific channels. (See the chip-specific configuration for availability.)
- Analog watchdogs allow continuous hardware monitoring of the converted value from input channels. (See the chip-specific configuration for availability.)
- Capacitive self test
- Presampling

36.2.3.2 Initiating conversions

Conversions can be initiated by either software or hardware.

The ADCD has an on-chip Cross-Triggering Unit (CTU) interface that can also initiate conversions. The CTU interface supports CTU Control mode.

An example of an application for which CTU triggering is suitable is a current control loop for lamps or LEDs. Using an on-chip timer module as trigger source, a CTU can initiate periodic conversions.

36.2.3.3 Interrupt/DMA support

The ADCD provides interrupt/DMA support for each type of channel for various end-of-channel conversion conditions. Data can be transferred via DMA.

36.2.3.4 Self-test

The ADCD can be configured to periodically check the health of the ADCA through various self-tests and communicate any critical/non-critical faults to an on-chip Fault Collection and Control Unit (FCCU) if available (See the chip-specific section for availability). The severity (critical/noncritical) of the different tests is programmable.

36.3 Features

The following general features are implemented in the ADC. (See the chip-specific section for availability of features, as each instance of an ADC within a chip may support different features.)

- Resolution: 12-bit
- Maximum speed: 1 Mega samples/second at 80 MHz
- 16 channels
- Sampling time register
  - *precision* channels:
    - 0 - 15 (ADC0)
    - 0 - 15 (ADC1)
    - 0 - 4, 10, 14 (ADC2)
    - 0 - 7, 10 - 14 (ADC3)

NOTE

The Temperature Sensor uses the value in the CTR1 register

- Modes of operation:
  - Normal
  - Injected
  - CTU
- Normal mode supports One-Shot/Scan (continuous)
- Injected mode supports One-Shot
- Presampling
- Power-Down mode
- Two abort features that allow you to abort either a single channel in a chain or the full chain in normal and injected mode
- Each channel has a dedicated register for conversion results and mode of operation (normal, injected, or CTU)
- Analog watchdog support:
  - Up to 16 analog watchdogs
  - Configurable for different channels

- Auto clock-off feature
- Programmable clock prescaler (bus clock divided by 2)
- CTU Control mode
- DMA support for each channel
- Interrupts for the following conditions:
  - End of conversion for a single channel for both normal and injected conversions
  - End of conversion for a chain for both normal and injected conversions
  - End of CTU conversion
  - Watchdog thresholds crossover
- Software-initiated calibration
- Self-testing feature

36.4 Memory Map/Register Definition

The following tables describe the registers and bit meanings for this module.

NOTE

There will be an Access Error for Reserved and out-of-range addresses. Special exceptions are the Reserved spaces at 039Ch and 03A4h: Do not write or change any value in these locations, or the ADC might malfunction.

NOTE

See the chip-specific configuration information because not all registers are available in all ADC instances.

Control logic registers:

- Main Configuration Register (ADC_MCR)
- Main Status register (ADC_MSR)
- Interrupt Status Register (ADC_ISR)
- Channel Pending register 0 (ADC_CEOCFR0)
- Interrupt Mask Register (ADC_IMR)
- Channel Interrupt Mask Register 0 (ADC_CIMR0)
- Watchdog Threshold Interrupt Status Register (ADC_WTISR)
- Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)
- Analog Watchdog Out of Range Register 0 (ADC_AWORR0)

DMA registers:

- DMA Enable register (ADC_DMAE)
- DMA Channel Select Register 0 (ADC_DMAR0)

Threshold registers:

These registers are used to store the user programmable lower and upper thresholds 12-bit values. These registers include:

- Threshold Register (ADC_THRHLR*n*)

Presampling registers:

- Presampling Control Register (ADC_PSCR)
- Presampling register 0 (ADC_PSR0)

Conversion timing registers:

- Conversion Timing Register 0 (ADC_CTR0)
- Conversion Timing Register 1 (ADC_CTR1)

Mask register: These registers are used to program which input channels must be converted during Normal and Injected conversion. These registers include:

- Normal Conversion Mask Register 0 (ADC_NCMR0)
- Injected Conversion Mask Register 0 (ADC_JCMR0)

Delay registers:

- Power Down Exit Delay Register (ADC_PDEDR)

Data registers:

The conversion results for the internal channels are loaded into data registers. Each data register also gives information regarding the corresponding result. These registers start with CDR0:

- Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)

Self test registers:

- Self Test Configuration Register 1 (ADC_STCR1)
- Self Test Configuration Register 2 (ADC_STCR2)
- Self Test Configuration Register 3 (ADC_STCR3)
- Self Test Status Register 1 (ADC_STSR1)
- Self Test Status Register 2 (ADC_STSR2)
- Self Test Status Register 3 (ADC_STSR3)
- Self Test Status Register 4 (ADC_STSR4)
- Self Test Baud Rate Register (ADC_STBRR)
- Self Test Analog Watchdog Register 0 (ADC_STAW0R)
- Self Test Analog Watchdog Register 4 (ADC_STAW4R)
- Self Test Analog Watchdog Register 1A (ADC_STAW1AR)
- Self Test Analog Watchdog Register 1B (ADC_STAW1BR)
- Self Test Analog Watchdog Register 2 (ADC_STAW2R)
- Self Test Analog Watchdog Register 5 (ADC_STAW5R)

- Self Test Data Register 1 (ADC_STDR1)
- Self Test Data Register 2 (ADC_STDR2)

Other registers include:

- Calibration, BIST Control and status Register (ADC_CALBISTREG)
- Offset and Gain User Register (ADC_OFSGNUSR)

ADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Main Configuration Register (ADC_MCR)	32	R/W	0000_0101h	36.4.1/974
4	Main Status register (ADC_MSR)	32	R	0000_0001h	36.4.2/977
10	Interrupt Status Register (ADC_ISR)	32	R/W	0000_0000h	36.4.3/979
14	Channel Pending register 0 (ADC_CEOCFR0)	32	R/W	0000_0000h	36.4.4/980
20	Interrupt Mask Register (ADC_IMR)	32	R/W	0000_0000h	36.4.5/982
24	Channel Interrupt Mask Register 0 (ADC_CIMR0)	32	R/W	0000_0000h	36.4.6/982
30	Watchdog Threshold Interrupt Status Register (ADC_WTISR)	32	R/W	0000_0000h	36.4.7/985
34	Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)	32	R/W	0000_0000h	36.4.8/988
40	DMA Enable register (ADC_DMAE)	32	R/W	0000_0000h	36.4.9/991
44	DMA Channel Select Register 0 (ADC_DMAR0)	32	R/W	0000_0000h	36.4.10/ 992
60	Threshold Register (ADC_THRHLR0)	32	R/W	0FFF_0000h	36.4.11/ 993
64	Threshold Register (ADC_THRHLR1)	32	R/W	0FFF_0000h	36.4.11/ 993
68	Threshold Register (ADC_THRHLR2)	32	R/W	0FFF_0000h	36.4.11/ 993
6C	Threshold Register (ADC_THRHLR3)	32	R/W	0FFF_0000h	36.4.11/ 993
80	Presampling Control Register (ADC_PSCR)	32	R/W	0000_0000h	36.4.12/ 994
84	Presampling register 0 (ADC_PSR0)	32	R/W	0000_0000h	36.4.13/ 995
94	Conversion Timing Register 0 (ADC_CTR0)	32	R/W	0000_0014h	36.4.14/ 996
98	Conversion Timing Register 1 (ADC_CTR1)	32	R/W	0000_0014h	36.4.15/ 997
A4	Normal Conversion Mask Register 0 (ADC_NCMR0)	32	R/W	0000_0000h	36.4.16/ 998
B4	Injected Conversion Mask Register 0 (ADC_JCMR0)	32	R/W	0000_0000h	36.4.17/ 999
C8	Power Down Exit Delay Register (ADC_PDEDR)	32	R/W	0000_0000h	36.4.18/ 1001
100	Channel Data Register n (Precision Channels) (ADC_CDR0)	32	R	0000_0000h	36.4.19/ 1002

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
104	Channel Data Register n (Precision Channels) (ADC_CDR1)	32	R	0000_0000h	36.4.19/ 1002
108	Channel Data Register n (Precision Channels) (ADC_CDR2)	32	R	0000_0000h	36.4.19/ 1002
10C	Channel Data Register n (Precision Channels) (ADC_CDR3)	32	R	0000_0000h	36.4.19/ 1002
110	Channel Data Register n (Precision Channels) (ADC_CDR4)	32	R	0000_0000h	36.4.19/ 1002
114	Channel Data Register n (Precision Channels) (ADC_CDR5)	32	R	0000_0000h	36.4.19/ 1002
118	Channel Data Register n (Precision Channels) (ADC_CDR6)	32	R	0000_0000h	36.4.19/ 1002
11C	Channel Data Register n (Precision Channels) (ADC_CDR7)	32	R	0000_0000h	36.4.19/ 1002
120	Channel Data Register n (Precision Channels) (ADC_CDR8)	32	R	0000_0000h	36.4.19/ 1002
124	Channel Data Register n (Precision Channels) (ADC_CDR9)	32	R	0000_0000h	36.4.19/ 1002
128	Channel Data Register n (Precision Channels) (ADC_CDR10)	32	R	0000_0000h	36.4.19/ 1002
12C	Channel Data Register n (Precision Channels) (ADC_CDR11)	32	R	0000_0000h	36.4.19/ 1002
130	Channel Data Register n (Precision Channels) (ADC_CDR12)	32	R	0000_0000h	36.4.19/ 1002
134	Channel Data Register n (Precision Channels) (ADC_CDR13)	32	R	0000_0000h	36.4.19/ 1002
138	Channel Data Register n (Precision Channels) (ADC_CDR14)	32	R	0000_0000h	36.4.19/ 1002
13C	Channel Data Register n (Precision Channels) (ADC_CDR15)	32	R	0000_0000h	36.4.19/ 1002
280	Threshold Register (ADC_THRHLR4)	32	R/W	0FFF_0000h	36.4.20/ 1003
284	Threshold Register (ADC_THRHLR5)	32	R/W	0FFF_0000h	36.4.20/ 1003
288	Threshold Register (ADC_THRHLR6)	32	R/W	0FFF_0000h	36.4.20/ 1003
28C	Threshold Register (ADC_THRHLR7)	32	R/W	0FFF_0000h	36.4.20/ 1003
290	Threshold Register (ADC_THRHLR8)	32	R/W	0FFF_0000h	36.4.20/ 1003
294	Threshold Register (ADC_THRHLR9)	32	R/W	0FFF_0000h	36.4.20/ 1003
298	Threshold Register (ADC_THRHLR10)	32	R/W	0FFF_0000h	36.4.20/ 1003

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29C	Threshold Register (ADC_THRHLR11)	32	R/W	0FFF_0000h	36.4.20/ 1003
2A0	Threshold Register (ADC_THRHLR12)	32	R/W	0FFF_0000h	36.4.20/ 1003
2A4	Threshold Register (ADC_THRHLR13)	32	R/W	0FFF_0000h	36.4.20/ 1003
2A8	Threshold Register (ADC_THRHLR14)	32	R/W	0FFF_0000h	36.4.20/ 1003
2AC	Threshold Register (ADC_THRHLR15)	32	R/W	0FFF_0000h	36.4.20/ 1003
2B0	Channel Watchdog Select Register 0 (ADC_CWSELR0)	32	R/W	0000_0000h	36.4.21/ 1003
2B4	Channel Watchdog Select Register 1 (ADC_CWSELR1)	32	R/W	0000_0000h	36.4.22/ 1004
2E0	Channel Watchdog Enable Register 0 (ADC_CWENR0)	32	R/W	0000_0000h	36.4.23/ 1005
2F0	Analog Watchdog Out of Range Register 0 (ADC_AWORR0)	32	R/W	0000_0000h	36.4.24/ 1007
340	Self Test Configuration Register 1 (ADC_STCR1)	32	R/W	1818_2507h	36.4.25/ 1009
344	Self Test Configuration Register 2 (ADC_STCR2)	32	R/W	0000_0005h	36.4.26/ 1010
348	Self Test Configuration Register 3 (ADC_STCR3)	32	R/W	0000_0300h	36.4.27/ 1012
34C	Self Test Baud Rate Register (ADC_STBRR)	32	R/W	0005_0000h	36.4.28/ 1013
350	Self Test Status Register 1 (ADC_STSR1)	32	R/W	0000_0000h	36.4.29/ 1015
354	Self Test Status Register 2 (ADC_STSR2)	32	R	0000_0000h	36.4.30/ 1018
358	Self Test Status Register 3 (ADC_STSR3)	32	R	0000_0000h	36.4.31/ 1019
35C	Self Test Status Register 4 (ADC_STSR4)	32	R	0000_0000h	36.4.32/ 1019
370	Self Test Data Register 1 (ADC_STDR1)	32	R	0000_0000h	36.4.33/ 1020
374	Self Test Data Register 2 (ADC_STDR2)	32	R	0000_0000h	36.4.34/ 1022
380	Self Test Analog Watchdog Register 0 (ADC_STAW0R)	32	R/W	0727_04C5h	36.4.35/ 1023
384	Self Test Analog Watchdog Register 1A (ADC_STAW1AR)	32	R/W	0003_0001h	36.4.36/ 1025
388	Self Test Analog Watchdog Register 1B (ADC_STAW1BR)	32	R/W	03E8_0ED0h	36.4.37/ 1026

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
38C	Self Test Analog Watchdog Register 2 (ADC_STAW2R)	32	R/W	0000_0FF9h	36.4.38/ 1027
394	Self Test Analog Watchdog Register 4 (ADC_STAW4R)	32	R/W	0010_0FF0h	36.4.39/ 1028
398	Self Test Analog Watchdog Register 5 (ADC_STAW5R)	32	R/W	0010_0FF0h	36.4.40/ 1029
3A0	Calibration, BIST Control and status Register (ADC_CALBISTREG)	32	R/W	C037_06F0h	36.4.41/ 1031
3A8	Offset and Gain User Register (ADC_OFSGNUSR)	32	R/W	0000_0000h	36.4.42/ 1033

36.4.1 Main Configuration Register (ADC_MCR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OWREN	WLSIDE	MODE	0	TRGEN	EDGE	Reserved	NSTART	0	JTRGEN	JEDGE	JSTART	0		CTUEN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	STCL	0						ADCLKSEL	ABORT_ CHAIN	ABORT	ACKO	0		REFSEL		PWDN
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

ADC_MCR field descriptions

Field	Description
0 OWREN	Overwrite enable. 0 Older valid conversion data is not overwritten by newer conversion data. 1 Newer conversion result is always overwritten, irrespective of the validity of older conversion data.
1 WLSIDE	Write Left/Right aligned. 0 The conversion data is written right aligned from (32-resolution) to 31). 1 Data is left aligned (from 16 to (16 + resolution - 1).
2 MODE	One_Shot/Scan.

Table continues on the next page...

ADC_MCR field descriptions (continued)

Field	Description
	0 One_Shot Mode: configure the Normal conversion of one chain. 1 Scan Mode: configure continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 TRGEN	External trigger enable. When set, enables the external trigger to start a conversion; if the external trigger feature is needed this bit must be set.
5 EDGE	Start trigger edge detection. If TRGEN is 1 this bit selects the falling (EDGE = zero) or rising (EDGE = one) edge for the external trigger.
6 Reserved	This field is reserved. **NOTE:** User must not overwrite default value of this field.
7 NSTART	Start of Normal conversion. Setting this field starts a normal conversion. In One shot mode: This field is cleared as soon as conversion is started. Setting this field during any ongoing conversion keeps this bit high until the requested conversion is started. **NOTE:** This bit cannot be set (1) when PWDN = 1. This bit is always cleared when PWDN = 1.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 JTRGEN	Injection external trigger enable. When set, enables the external trigger to start a injected conversion; if the external trigger feature is needed for injected conversion then this field must be set.
10 JEDGE	Injection trigger edge selection. Edge selection for external trigger; if JTRGEN is one this bit selects the falling (JEDGE = zero) or rising (JEDGE = one) edge for the external trigger.
11 JSTART	Injection start. Setting this field will start the configured injected conversion chain. Resetting this field has no effect, as the injected chain conversion cannot be interrupted. Setting this field during ongoing Injected conversion keeps this field high value until the requested conversion is started.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CTUEN	Crosstrigger Unit Enable. Enables the crosstriggering unit (CTU). 0 The crosstriggering unit is disabled and the CTU-triggered injected conversion cannot take place. 1 The crosstriggering unit is enabled and the CTUtriggered injected conversion can take place.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 STCL	Self Testing Configuration Lock.

Table continues on the next page...

ADC_MCR field descriptions (continued)

Field	Description
	0 No lock 1 The self-testing configuration is locked i.e. STCR1, STCR2, STCR3, STBRR, STAW0R, STAW1AR, STAW1BR, STAW2R, STAW4R, STAW5R are write-protected. It can be used only in CPU and SCAN mode. The lock bit is cleared only by a peripheral reset.
17–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 ADCLKSEL	Analog Clock frequency Selector. This field can be written in Power Down mode only. Please check device specific additional restriction on programming of this field. 0 ADC clock frequency is half of bus clock 1 ADC clock frequency is equal to bus clock
24 ABORT_CHAIN	Abort Chain. If this bit is set the ongoing Chain Conversion is aborted. This bit gets reset by hardware as soon as a new conversion is requested. **NOTE:** Setting this bit in idle state (no normal/injected conversion is ongoing) has no effect. In this case it is reset immediately. During ongoing CTU conversion this bit also cannot be programmed.
25 ABORT	Abort Conversion. If this bit is set, the ongoing channel conversion is aborted and the next channel conversion is invoked. This bit gets reset by hardware as soon as the new conversion is invoked. This bit cannot be written while self test conversion is ongoing. **NOTE:** Setting this bit in idle state (No Normal/injected conversion is ongoing) has no effect. In this case it is reset immediately. During ongoing CTU conversion this bit cannot be programmed
26 ACKO	Auto clock off enable. If set, enables the Auto clock off feature.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 REFSEL	Reference voltage selection for ADC analog part. 00 Select VREFH as reference voltage 01 Reserved 10 Reserved 11 Reserved
31 PWDN	Power-down enable. When this bit is set, the analog module is requested to enter Power-Down mode. When the ADC's status is PWDN, resetting this bit will start the ADC's transition to IDLE mode. In CTU control mode MCR[CTUEN] must be reset before entering into power-down mode.

36.4.2 Main Status register (ADC_MSR)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CALIBRTD	0						NSTART	JABORT	0		JSTART	0	SELF_TEST_S	0	CTUSTART
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHADDR							0			ACKO	0		ADCSTATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

ADC_MSR field descriptions

Field	Description
0 CALIBRTD	This bit indicates the ADC calibration status. This bit gets updated after running the High Accuracy Calibration mode.

Table continues on the next page...

ADC_MSR field descriptions (continued)

Field	Description
	0 Uncalibrated or calibration unsuccessful 1 Calibration was run and it was successful
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 NSTART	This status bit is used to signal that a Normal conversion is ongoing.
8 JABORT	This status bit is used to signal that an Injected conversion has been aborted by CTU. That bit is reset when a new injected conversion starts.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 JSTART	This status bit is used to signal that an Injected conversion is ongoing.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SELF_TEST_S	This status bit signals that self test conversion is ongoing.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CTUSTART	This status bit is used to signal that a CTU conversion is ongoing. This bit is set when a CTU trigger pulse is received and the CTU conversion starts.When Control Mode is enabled this bit is reset when the CTU is disabled (CTUEN set to 0).
16–22 CHADDR	Channel under measure address. Status bits are used to signal which channel is under measure.
23–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 ACKO	Auto clock off enable. This status bit is used to signal if the Auto clock off feature is on.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 ADCSTATUS	Status of the ADC. 000 Idle 001 Power Down 100 Sample 110 Conversion 010 Wait state (waiting to start conversion [external trigger]) 011 Busy in Calibration

36.4.3 Interrupt Status Register (ADC_ISR)

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											EOCTU	JEOC	JECH	EOC	ECH
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_ISR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 EOCTU	End of CTU conversion. This bit is set for a digital end of conversion on a the CTU Channel; active when set.
28 JEOC	End of injected channel conversion. This bit is set for a digital end of conversion on an injected channel; active when set.
29 JECH	End of injected chain conversion. This bit is set for a digital end of chain conversion on an injected channel; active when set. **NOTE:** If there is no channel selected in the ADC_JCMR*n* register(s) and there is a start of conversion trigger, then ADC_ISR[JECH] is set and a JECH interrupt is immediately issued (if enabled). Similarly, if no channel is selected in the ADC_NCMR*n* register and a normal conversion is triggered, ADC_ISR[ECH] is set and an ECH interrupt is immediately issued, if enabled.
30 EOC	End of channel conversion. This bit is set for a digital end of conversion on a Normal Channel; active when set.
31 ECH	End of chain conversion. This bit is set for a digital end of chain conversion on a Normal Channel; active when set. **NOTE:** If there is no channel selected in the ADC_NCMR*n* register(s) and there is a start of conversion trigger, then ADC_ISR[ECH] is set and an ECH interrupt is immediately issued (if enabled).

Table continues on the next page...

ADC_ISR field descriptions (continued)

Field	Description
	Similarly, if no channel is selected in the ADC_JCMR*n* register and an injected conversion is triggered, ADC_ISR[JECH] is set and a JECH interrupt is immediately issued, if enabled.

36.4.4 Channel Pending register 0 (ADC_CEOCFR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOCCH15	EOCCH14	EOCCH13	EOCCH12	EOCCH11	EOCCH10	EOCCH9	EOCCH8	EOCCH7	EOCCH6	EOCCH5	EOCCH4	EOCCH3	EOCCH2	EOCCH1	EOCCH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CEOCFR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EOCCH15	EOC Channel 15. When set, the measure of channel 15 is completed.

Table continues on the next page...

ADC_CEOCFR0 field descriptions (continued)

Field	Description
17 EOCCH14	EOC Channel 14. When set, the measure of channel 14 is completed.
18 EOCCH13	EOC Channel 13. When set, the measure of channel 13 is completed.
19 EOCCH12	EOC Channel 12. When set, the measure of channel 12 is completed.
20 EOCCH11	EOC Channel 11. When set, the measure of channel 11 is completed.
21 EOCCH10	EOC Channel 10. When set, the measure of channel 10 is completed.
22 EOCCH9	EOC Channel 9. When set, the measure of channel 9 is completed.
23 EOCCH8	EOC Channel 8. When set, the measure of channel 8 is completed.
24 EOCCH7	EOC Channel 7. When set, the measure of channel 7 is completed.
25 EOCCH6	EOC Channel 6. When set, the measure of channel 6 is completed.
26 EOCCH5	EOC Channel 5. When set, the measure of channel 5 is completed.
27 EOCCH4	EOC Channel 4. When set, the measure of channel 4 is completed.
28 EOCCH3	EOC Channel 3. When set, the measure of channel 3 is completed.
29 EOCCH2	EOC Channel 2. When set, the measure of channel 2 is completed.
30 EOCCH1	EOC Channel 1. When set, the measure of channel 1 is completed.
31 EOCCH0	EOC Channel 0. When set, the measure of channel 0 is completed.

36.4.5 Interrupt Mask Register (ADC_IMR)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	0										MSKEOCTU	MSKJEOC	MSKJECH	MSKEOC	MSKECH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_IMR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. Reserved
17–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 MSKEOCTU	Mask bit for EOCTU. When set, the interrupt is enabled.
28 MSKJEOC	Mask bit for JEOC. When set, the interrupt is enabled.
29 MSKJECH	Mask bit for JECH. When set, the interrupt is enabled.
30 MSKEOC	Mask bit for EOC. When set, the interrupt is enabled.
31 MSKECH	Mask bit for ECH. When set, the interrupt is enabled.

36.4.6 Channel Interrupt Mask Register 0 (ADC_CIMR0)

The fields in this register globally enable or disable end of conversion interrupts by channels within the range of Channel 0 through 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific details to find which channels are applicable.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIM15	CIM14	CIM13	CIM12	CIM11	CIM10	CIM9	CIM8	CIM7	CIM6	CIM5	CIM4	CIM3	CIM2	CIM1	CIM0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CIMR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CIM15	Channel 15 interrupt enable. When this field is set, the interrupt for channel 15 is enabled.
17 CIM14	Channel 14 interrupt enable. When this field is set, the interrupt for channel 14 is enabled.
18 CIM13	Channel 13 interrupt enable. When this field is set, the interrupt for channel 13 is enabled.
19 CIM12	Channel 12 interrupt enable. When this field is set, the interrupt for channel 12 is enabled.
20 CIM11	Channel 11 interrupt enable. When this field is set, the interrupt for channel 11 is enabled.
21 CIM10	Channel 10 interrupt enable. When this field is set, the interrupt for channel 10 is enabled.
22 CIM9	Channel 9 interrupt enable. When this field is set, the interrupt for channel 9 is enabled.
23 CIM8	Channel 8 interrupt enable. When this field is set, the interrupt for channel 8 is enabled.
24 CIM7	Channel 7 interrupt enable. When this field is set, the interrupt for channel 7 is enabled.

Table continues on the next page...

ADC_CIMR0 field descriptions (continued)

Field	Description
25 CIM6	Channel 6 interrupt enable. When this field is set, the interrupt for channel 6 is enabled.
26 CIM5	Channel 5 interrupt enable. When this field is set, the interrupt for channel 5 is enabled.
27 CIM4	Channel 4 interrupt enable. When this field is set, the interrupt for channel 4 is enabled.
28 CIM3	Channel 3 interrupt enable. When this field is set, the interrupt for channel 3 is enabled.
29 CIM2	Channel 2 interrupt enable. When this field is set, the interrupt for channel 2 is enabled.
30 CIM1	Channel 1 interrupt enable. When this field is set, the interrupt for channel 1 is enabled.
31 CIM0	Channel 0 interrupt enable. When this field is set, the interrupt for channel 0 is enabled.

36.4.7 Watchdog Threshold Interrupt Status Register (ADC_WTISR)

The fields in this register, when set, indicate either the upper or lower threshold value has been crossed for a specific watchdog.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_WTISR field descriptions

Field	Description
0 WDG15H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 15 (WD15). **NOTE:** Write a '1' to this field to clear it.
1 WDG15L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 15 (WD15). **NOTE:** Write a '1' to this field to clear it.
2 WDG14H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 14 (WD14). **NOTE:** Write a '1' to this field to clear it.
3 WDG14L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 14 (WD14). **NOTE:** Write a '1' to this field to clear it.

Table continues on the next page...

ADC_WTISR field descriptions (continued)

Field	Description
4 WDG13H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 13 (WD13). **NOTE:** Write a '1' to this field to clear it.
5 WDG13L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 13 (WD13). **NOTE:** Write a '1' to this field to clear it.
6 WDG12H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 12 (WD12). **NOTE:** Write a '1' to this field to clear it.
7 WDG12L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 12 (WD12). **NOTE:** Write a '1' to this field to clear it.
8 WDG11H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 11 (WD11). **NOTE:** Write a '1' to this field to clear it.
9 WDG11L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 11 (WD11). **NOTE:** Write a '1' to this field to clear it.
10 WDG10H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 10 (WD10). **NOTE:** Write a '1' to this field to clear it.
11 WDG10L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 10 (WD10). **NOTE:** Write a '1' to this field to clear it.
12 WDG9H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 9 (WD9). **NOTE:** Write a '1' to this field to clear it.
13 WDG9L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 9 (WD9). **NOTE:** Write a '1' to this field to clear it.
14 WDG8H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 8 (WD8). **NOTE:** Write a '1' to this field to clear it.
15 WDG8L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 8 (WD8). **NOTE:** Write a '1' to this field to clear it.
16 WDG7H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 7 (WD7). **NOTE:** Write a '1' to this field to clear it.

Table continues on the next page...

ADC_WTISR field descriptions (continued)

Field	Description
17 WDG7L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 7 (WD7). **NOTE:** Write a '1' to this field to clear it.
18 WDG6H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 6 (WD6). **NOTE:** Write a '1' to this field to clear it.
19 WDG6L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 6 (WD6). **NOTE:** Write a '1' to this field to clear it.
20 WDG5H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 5 (WD5). **NOTE:** Write a '1' to this field to clear it.
21 WDG5L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 5 (WD5). **NOTE:** Write a '1' to this field to clear it.
22 WDG4H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 4 (WD4). **NOTE:** Write a '1' to this field to clear it.
23 WDG4L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 4 (WD4). **NOTE:** Write a '1' to this field to clear it.
24 WDG3H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 3 (WD3). **NOTE:** Write a '1' to this field to clear it.
25 WDG3L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 3 (WD3). **NOTE:** Write a '1' to this field to clear it.
26 WDG2H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 2 (WD2). **NOTE:** Write a '1' to this field to clear it.
27 WDG2L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 2 (WD2). **NOTE:** Write a '1' to this field to clear it.
28 WDG1H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 1 (WD1). **NOTE:** Write a '1' to this field to clear it.
29 WDG1L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 1 (WD1). **NOTE:** Write a '1' to this field to clear it.

Table continues on the next page...

ADC_WTISR field descriptions (continued)

Field	Description
30 WDG0H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 0 (WD0). **NOTE:** Write a '1' to this field to clear it.
31 WDG0L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 0 (WD0). **NOTE:** Write a '1' to this field to clear it.

36.4.8 Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_WTIMR field descriptions

Field	Description
0 MSKWDG15H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 15 (WD15) being higher than the programmed higher threshold. When set, the interrupt is enabled.
1 MSKWDG15L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 15 (WD15) being lower than the programmed lower threshold. When set, the interrupt is enabled.
2 MSKWDG14H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 14 (WD14) being higher than the programmed higher threshold. When set, the interrupt is enabled.
3 MSKWDG14L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 14 (WD14) being lower than the programmed lower threshold. When set, the interrupt is enabled.
4 MSKWDG13H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 13 (WD13) being higher than the programmed higher threshold. When set, the interrupt is enabled.

Table continues on the next page...

ADC_WTIMR field descriptions (continued)

Field	Description
5 MSKWDG13L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 13 (WD13) being lower than the programmed lower threshold. When set, the interrupt is enabled.
6 MSKWDG12H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 12 (WD12) being higher than the programmed higher threshold. When set, the interrupt is enabled.
7 MSKWDG12L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 12 (WD12) being lower than the programmed lower threshold. When set, the interrupt is enabled.
8 MSKWDG11H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 11 (WD11) being higher than the programmed higher threshold. When set, the interrupt is enabled.
9 MSKWDG11L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 11 (WD11) being lower than the programmed lower threshold. When set, the interrupt is enabled.
10 MSKWDG10H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 10 (WD10) being higher than the programmed higher threshold. When set, the interrupt is enabled.
11 MSKWDG10L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 10 (WD10) being lower than the programmed lower threshold. When set, the interrupt is enabled.
12 MSKWDG9H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 9 (WD9) being higher than the programmed higher threshold. When set, the interrupt is enabled.
13 MSKWDG9L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 9 (WD9) being lower than the programmed lower threshold. When set, the interrupt is enabled.
14 MSKWDG8H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 8 (WD8) being higher than the programmed higher threshold. When set, the interrupt is enabled.
15 MSKWDG8L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 8 (WD8) being lower than the programmed lower threshold. When set, the interrupt is enabled.
16 MSKWDG7H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 7 (WD7) being higher than the programmed higher threshold. When set, the interrupt is enabled.
17 MSKWDG7L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 7 (WD7) being lower than the programmed lower threshold. When set, the interrupt is enabled.
18 MSKWDG6H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 6 (WD6) being higher than the programmed higher threshold. When set, the interrupt is enabled.
19 MSKWDG6L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 6 (WD6) being lower than the programmed lower threshold. When set, the interrupt is enabled.
20 MSKWDG5H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 5 (WD5) being higher than the programmed higher threshold. When set, the interrupt is enabled.

Table continues on the next page...

ADC_WTIMR field descriptions (continued)

Field	Description
21 MSKWDG5L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 5 (WD5) being lower than the programmed lower threshold. When set, the interrupt is enabled.
22 MSKWDG4H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 4 (WD4) being higher than the programmed higher threshold. When set, the interrupt is enabled.
23 MSKWDG4L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 4 (WD4) being lower than the programmed lower threshold. When set, the interrupt is enabled.
24 MSKWDG3H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 3 (WD3) being higher than the programmed higher threshold. When set, the interrupt is enabled.
25 MSKWDG3L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 3 (WD3) being lower than the programmed lower threshold. When set, the interrupt is enabled.
26 MSKWDG2H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 2 (WD2) being higher than the programmed higher threshold. When set, the interrupt is enabled.
27 MSKWDG2L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 2 (WD2) being lower than the programmed lower threshold. When set, the interrupt is enabled.
28 MSKWDG1H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 1 (WD1) being higher than the programmed higher threshold. When set, the interrupt is enabled.
29 MSKWDG1L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 1 (WD1) being lower than the programmed lower threshold. When set, the interrupt is enabled.
30 MSKWDG0H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 0 (WD0) being higher than the programmed higher threshold. When set, the interrupt is enabled.
31 MSKWDG0L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 0 (WD0) being lower than the programmed lower threshold. When set, the interrupt is enabled.

36.4.9 DMA Enable register (ADC_DMAE)

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														DCLR	DMAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DCLR	DMA Clear sequence enable. 0 DMA request cleared by Acknowledge from DMA controller 1 DMA request cleared on read of data registers **NOTE:** When DCLR is set, ADC DMA channels should only execute read accesses on ADC_CDR*n* registers.
31 DMAEN	DMA global enable. When this bit is set, the DMA feature is enabled.

36.4.10 DMA Channel Select Register 0 (ADC_DMAR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA15	DMA14	DMA13	DMA12	DMA11	DMA10	DMA9	DMA8	DMA7	DMA6	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 DMA15	DMA enable for channel 15. When set, transferring data in DMA mode is enabled for channel 15.
17 DMA14	DMA enable for channel 14. When set, transferring data in DMA mode is enabled for channel 14.
18 DMA13	DMA enable for channel 13. When set, transferring data in DMA mode is enabled for channel 13.
19 DMA12	DMA enable for channel 12. When set, transferring data in DMA mode is enabled for channel 12.
20 DMA11	DMA enable for channel 11. When set, transferring data in DMA mode is enabled for channel 11.
21 DMA10	DMA enable for channel 10. When set, transferring data in DMA mode is enabled for channel 10.
22 DMA9	DMA enable for channel 9.

Table continues on the next page...

ADC_DMAR0 field descriptions (continued)

Field	Description
	When set, transferring data in DMA mode is enabled for channel 9.
23 DMA8	DMA enable for channel 8. When set, transferring data in DMA mode is enabled for channel 8.
24 DMA7	DMA enable for channel 7. When set, transferring data in DMA mode is enabled for channel 7.
25 DMA6	DMA enable for channel 6. When set, transferring data in DMA mode is enabled for channel 6.
26 DMA5	DMA enable for channel 5. When set, transferring data in DMA mode is enabled for channel 5.
27 DMA4	DMA enable for channel 4. When set, transferring data in DMA mode is enabled for channel 4.
28 DMA3	DMA enable for channel 3. When set, transferring data in DMA mode is enabled for channel 3.
29 DMA2	DMA enable for channel 2. When set, transferring data in DMA mode is enabled for channel 2.
30 DMA1	DMA enable for channel 1. When set, transferring data in DMA mode is enabled for channel 1.
31 DMA0	DMA enable for channel 0. When set, transferring data in DMA mode is enabled for channel 0.

36.4.11 Threshold Register (ADC_THRHLR*n*)

Address: 0h base + 60h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRH												0				THRL											
W	▨	▨	▨	▨													▨	▨	▨	▨												
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_THRHLR*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for watchdog n.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for watchdog n.

36.4.12 Presampling Control Register (ADC_PSCR)

See the chip configuration details for the availability of this register.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									Reserved		Reserved		PREVAL0		PRECONV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSCR field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–26 Reserved	This field is reserved.
27–28 Reserved	This field is reserved.
29–30 PREVAL0	Internal voltage selection for Presampling. These two bits select analog input voltage for presampling from the available four internal voltages for Internal precision channels (Channel 0 to 31). Refer to Presampling channel enable.
31 PRECONV	Convert Presampled value If bit PRECONV is set, presampling is followed by the conversion. Sampling will be bypassed and conversion of presampled data will be done. This bit has no effect on conversion of a channel when presampling on that channel is disabled (for examplei.e PSR*x*[PRES*y*] = 0 , where *x* = register index and *y* = channel number).

36.4.13 Presampling register 0 (ADC_PSR0)

See the chip configuration details for the availability of this register.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRES15	PRES14	PRES13	PRES12	PRES11	PRES10	PRES9	PRES8	PRES7	PRES6	PRES5	PRES4	PRES3	PRES2	PRES1	PRES0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 PRES15	Presampling enable for channel 15. When set, presampling is enabled for channel 15.
17 PRES14	Presampling enable for channel 14. When set, presampling is enabled for channel 14.
18 PRES13	Presampling enable for channel 13. When set, presampling is enabled for channel 13.
19 PRES12	Presampling enable for channel 12. When set, presampling is enabled for channel 12.
20 PRES11	Presampling enable for channel 11. When set, presampling is enabled for channel 11.
21 PRES10	Presampling enable for channel 10. When set, presampling is enabled for channel 10.
22 PRES9	Presampling enable for channel 9. When set, presampling is enabled for channel 9.
23 PRES8	Presampling enable for channel 8. When set, presampling is enabled for channel 8.

Table continues on the next page...

ADC_PSR0 field descriptions (continued)

Field	Description
24 PRES7	Presampling enable for channel 7. When set, presampling is enabled for channel 7.
25 PRES6	Presampling enable for channel 6. When set, presampling is enabled for channel 6.
26 PRES5	Presampling enable for channel 5. When set, presampling is enabled for channel 5.
27 PRES4	Presampling enable for channel 4. When set, presampling is enabled for channel 4.
28 PRES3	Presampling enable for channel 3. When set, presampling is enabled for channel 3.
29 PRES2	Presampling enable for channel 2. When set, presampling is enabled for channel 2.
30 PRES1	Presampling enable for channel 1. When set, presampling is enabled for channel 1.
31 PRES0	Presampling enable for channel 0. When set, presampling is enabled for channel 0.

36.4.14 Conversion Timing Register 0 (ADC_CTR0)

Conversion timing register. CTR0 is associated with channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	0	Reserved		0	Reserved		0	INPSAMP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

ADC_CTR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–19 Reserved	This field is reserved.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 Reserved	This field is reserved.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles. **NOTE:** The Bandgap voltage sampling time is controlled by CTR0[INPSAMP]. CTR0[INPSAMP] is also used for the external channel sample time. When the user software needs to sample the Bandgap voltage, it must change the CTR0[INPSAMP] value to allow for a longer time while the Bandgap is being sampled, then change its value back to the normal value for sampling external channels.

36.4.15 Conversion Timing Register 1 (ADC_CTR1)

Please see the chip-specific configuration information as all channels may not be available in all ADC instances.

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								INPSAMP							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

ADC_CTR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles.

36.4.16 Normal Conversion Mask Register 0 (ADC_NCMR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_NCMR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CH15	Normal sampling enable for channel 15. When set, normal sampling is enabled for channel 15.
17 CH14	Normal sampling enable for channel 14. When set, normal sampling is enabled for channel 14.
18 CH13	Normal sampling enable for channel 13. When set, normal sampling is enabled for channel 13.

Table continues on the next page...

ADC_NCMR0 field descriptions (continued)

Field	Description
19 CH12	Normal sampling enable for channel 12. When set, normal sampling is enabled for channel 12.
20 CH11	Normal sampling enable for channel 11. When set, normal sampling is enabled for channel 11.
21 CH10	Normal sampling enable for channel 10. When set, normal sampling is enabled for channel 10.
22 CH9	Normal sampling enable for channel 9. When set, normal sampling is enabled for channel 9.
23 CH8	Normal sampling enable for channel 8. When set, normal sampling is enabled for channel 8.
24 CH7	Normal sampling enable for channel 7. When set, normal sampling is enabled for channel 7.
25 CH6	Normal sampling enable for channel 6. When set, normal sampling is enabled for channel 6.
26 CH5	Normal sampling enable for channel 5. When set, normal sampling is enabled for channel 5.
27 CH4	Normal sampling enable for channel 4. When set, normal sampling is enabled for channel 4.
28 CH3	Normal sampling enable for channel 3. When set, normal sampling is enabled for channel 3.
29 CH2	Normal sampling enable for channel 2. When set, normal sampling is enabled for channel 2.
30 CH1	Normal sampling enable for channel 1. When set, normal sampling is enabled for channel 1.
31 CH0	Normal sampling enable for channel 0. When set, normal sampling is enabled for channel 0.

36.4.17 Injected Conversion Mask Register 0 (ADC_JCMR0)

Enable bits of Injected Sampling for channel 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_JCMR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CH15	Injected sampling enable for channel 15. When set, injected sampling is enabled for channel 15.
17 CH14	Injected sampling enable for channel 14. When set, injected sampling is enabled for channel 14.
18 CH13	Injected sampling enable for channel 13. When set, injected sampling is enabled for channel 13.
19 CH12	Injected sampling enable for channel 12. When set, injected sampling is enabled for channel 12.
20 CH11	Injected sampling enable for channel 11. When set, injected sampling is enabled for channel 11.
21 CH10	Injected sampling enable for channel 10. When set, injected sampling is enabled for channel 10.
22 CH9	Injected sampling enable for channel 9. When set, injected sampling is enabled for channel 9.
23 CH8	Injected sampling enable for channel 8. When set, injected sampling is enabled for channel 8.
24 CH7	Injected sampling enable for channel 7. When set, injected sampling is enabled for channel 7.
25 CH6	Injected sampling enable for channel 6. When set, injected sampling is enabled for channel 6.
26 CH5	Injected sampling enable for channel 5. When set, injected sampling is enabled for channel 5.

Table continues on the next page...

ADC_JCMR0 field descriptions (continued)

Field	Description
27 CH4	Injected sampling enable for channel 4. When set, injected sampling is enabled for channel 4.
28 CH3	Injected sampling enable for channel 3. When set, injected sampling is enabled for channel 3.
29 CH2	Injected sampling enable for channel 2. When set, injected sampling is enabled for channel 2.
30 CH1	Injected sampling enable for channel 1. When set, injected sampling is enabled for channel 1.
31 CH0	Injected sampling enable for channel 0. When set, injected sampling is enabled for channel 0.

36.4.18 Power Down Exit Delay Register (ADC_PDEDR)

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								PDED							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PDEDR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PDED	The delay between the power down bit reset and the start of conversion. The power down delay is calculated as (PDED x 1/[ADC_clock_frequency]).

36.4.19 Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)

Note

It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

Address: 0h base + 100h offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CDR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table continues on the next page...

ADC_CDR*n* field descriptions (continued)

Field	Description
13 OVERW	Overwrite data Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the OWREN bit of MCR register.
14–15 RESULT	These two bits reflect the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16–31 CDATA	Converted Data 11:0. **Note:** It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

36.4.20 Threshold Register (ADC_THRHLR*n*)

Address: 0h base + 280h offset + (4d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRH												0				THRL											
W																																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_THRHLR*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for channel x.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for channel x.

36.4.21 Channel Watchdog Select Register 0 (ADC_CWSELR0)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH7				WSEL_CH6				WSEL_CH5				WSEL_CH4				WSEL_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR0 field descriptions

Field	Description
0–3 WSEL_CH7	Channel Watchdog select for channel 7. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH6	Channel Watchdog select for channel 6. See WSEL_CH7.
8–11 WSEL_CH5	Channel Watchdog select for channel 5. See WSEL_CH7.
12–15 WSEL_CH4	Channel Watchdog select for channel 4. See WSEL_CH7.
16–19 WSEL_CH3	Channel Watchdog select for channel 3. See WSEL_CH7.
20–23 WSEL_CH2	Channel Watchdog select for channel 2. See WSEL_CH7.
24–27 WSEL_CH1	Channel Watchdog select for channel 1. See WSEL_CH7.
28–31 WSEL_CH0	Channel Watchdog select for channel 0. See WSEL_CH7.

36.4.22 Channel Watchdog Select Register 1 (ADC_CWSELR1)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH15				WSEL_CH14				WSEL_CH13				WSEL_CH12				WSEL_CH11				WSEL_CH10				WSEL_CH9				WSEL_CH8			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR1 field descriptions

Field	Description
0–3 WSEL_CH15	Channel Watchdog select for channel 15. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH14	Channel Watchdog select for channel 14. See WSEL_CH15.
8–11 WSEL_CH13	Channel Watchdog select for channel 13. See WSEL_CH15.
12–15 WSEL_CH12	Channel Watchdog select for channel 12. See WSEL_CH15.
16–19 WSEL_CH11	Channel Watchdog select for channel 11. See WSEL_CH15.
20–23 WSEL_CH10	Channel Watchdog select for channel 10. See WSEL_CH15.
24–27 WSEL_CH9	Channel Watchdog select for channel 9. See WSEL_CH15.
28–31 WSEL_CH8	Channel Watchdog select for channel 8. See WSEL_CH15.

36.4.23 Channel Watchdog Enable Register 0 (ADC_CWENR0)

Controls whether the watchdog is enabled for channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CWEN15	CWEN14	CWEN13	CWEN12	CWEN11	CWEN10	CWEN9	CWEN8	CWEN7	CWEN6	CWEN5	CWEN4	CWEN3	CWEN2	CWEN1	CWEN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWENR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CWEN15	Watchdog enable for channel 15. When set, Watchdog feature is enabled for channel 15.
17 CWEN14	Watchdog enable for channel 14. When set, Watchdog feature is enabled for channel 14.
18 CWEN13	Watchdog enable for channel 13. When set, Watchdog feature is enabled for channel 13.
19 CWEN12	Watchdog enable for channel 12. When set, Watchdog feature is enabled for channel 12.
20 CWEN11	Watchdog enable for channel 11. When set, Watchdog feature is enabled for channel 11.
21 CWEN10	Watchdog enable for channel 10. When set, Watchdog feature is enabled for channel 10.
22 CWEN9	Watchdog enable for channel 9. When set, Watchdog feature is enabled for channel 9.
23 CWEN8	Watchdog enable for channel 8. When set, Watchdog feature is enabled for channel 8.
24 CWEN7	Watchdog enable for channel 7. When set, Watchdog feature is enabled for channel 7.

Table continues on the next page...

ADC_CWENR0 field descriptions (continued)

Field	Description
25 CWEN6	Watchdog enable for channel 6. When set, Watchdog feature is enabled for channel 6.
26 CWEN5	Watchdog enable for channel 5. When set, Watchdog feature is enabled for channel 5.
27 CWEN4	Watchdog enable for channel 4. When set, Watchdog feature is enabled for channel 4.
28 CWEN3	Watchdog enable for channel 3. When set, Watchdog feature is enabled for channel 3.
29 CWEN2	Watchdog enable for channel 2. When set, Watchdog feature is enabled for channel 2.
30 CWEN1	Watchdog enable for channel 1. When set, Watchdog feature is enabled for channel 1.
31 CWEN0	Watchdog enable for channel 0. When set, Watchdog feature is enabled for channel 0.

36.4.24 Analog Watchdog Out of Range Register 0 (ADC_AWORR0)

Analog watchdog out of range register for channels 0 to 31.

NOTE

> This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2F0h offset = 2F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	AWOR_CH9	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_AWORR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 AWOR_CH15	Analog watchdog out of range for channel 15. When set, indicates the converted data is out of range for channel 15.
17 AWOR_CH14	Analog watchdog out of range for channel 14. When set, indicates the converted data is out of range for channel 14.
18 AWOR_CH13	Analog watchdog out of range for channel 13. When set, indicates the converted data is out of range for channel 13.
19 AWOR_CH12	Analog watchdog out of range for channel 12. When set, indicates the converted data is out of range for channel 12.
20 AWOR_CH11	Analog watchdog out of range for channel 11. When set, indicates the converted data is out of range for channel 11.
21 AWOR_CH10	Analog watchdog out of range for channel 10. When set, indicates the converted data is out of range for channel 10.
22 AWOR_CH9	Analog watchdog out of range for channel 9. When set, indicates the converted data is out of range for channel 9.
23 AWOR_CH8	Analog watchdog out of range for channel 8. When set, indicates the converted data is out of range for channel 8.
24 AWOR_CH7	Analog watchdog out of range for channel 7. When set, indicates the converted data is out of range for channel 7.
25 AWOR_CH6	Analog watchdog out of range for channel 6. When set, indicates the converted data is out of range for channel 6.
26 AWOR_CH5	Analog watchdog out of range for channel 5. When set, indicates the converted data is out of range for channel 5.
27 AWOR_CH4	Analog watchdog out of range for channel 4. When set, indicates the converted data is out of range for channel 4.

Table continues on the next page...

ADC_AWORR0 field descriptions (continued)

Field	Description
28 AWOR_CH3	Analog watchdog out of range for channel 3. When set, indicates the converted data is out of range for channel 3.
29 AWOR_CH2	Analog watchdog out of range for channel 2. When set, indicates the converted data is out of range for channel 2.
30 AWOR_CH1	Analog watchdog out of range for channel 1. When set, indicates the converted data is out of range for channel 1.
31 AWOR_CH0	Analog watchdog out of range for channel 0. When set, indicates the converted data is out of range for channel 0.

36.4.25 Self Test Configuration Register 1 (ADC_STCR1)

Address: 0h base + 340h offset = 340h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INPSAMP_C								Reserved								INPSAMP_S								Reserved							
Reset	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	1	1

ADC_STCR1 field descriptions

Field	Description
0–7 INPSAMP_C	Sampling phase duration for the test conversions related to self test algorithm C. Minimum value should be set as per default (18h). Maximum can be FFh.
8–15 Reserved	This field is reserved.
16–23 INPSAMP_S	Sampling phase duration for the test conversions related to self test algorithm S. Minimum value should be set as per default (25h). Maximum can be FFh.
24–31 Reserved	This field is reserved.

36.4.26 Self Test Configuration Register 2 (ADC_STCR2)

Address: 0h base + 344h offset = 344h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				MSKWDSERR		MSKWDTERR	0	MSKST_EOC	Reserved				MSKWDG_EOA_C	Reserved	MSKWDG_EOA_S
W						SERR										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSKERR_C	Reserved	MSKERR_S2	MSKERR_S1	MSKERR_S0	0			EN	0		FMA_WDSERR	FMA_WDTERR	FMA_C	Reserved	FMA_S
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

ADC_STCR2 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MSKWDSERR	Interrupt enable (STSR1[WDSERR] status bit) 0 Interrupt disabled 1 Enables the STSR1[WDSERR] status bit to generate an interrupt indication.
5 SERR	Error fault injection bit (write only). Writing 1 to this bit sets the STSR1[ERRx] status bits once only. Reading always returns 0. Setting this bit does not affect any of the self test data registers.
6 MSKWDTERR	Interrupt enable (STSR[WDTERR] status bit) 0 Interrupt disabled 1 Enables the STSR1[WDTERR] status bit to generate an interrupt indication.

Table continues on the next page...

ADC_STCR2 field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MSKST_EOC	Interrupt Enable bit for STSR2[ST_EOC] 0 Interrupt disabled 1 Enables the STSR1[ST_EOC] status bit to generate an interrupt indication. (IMR[MSKEOC] must also be enabled.)
9–12 Reserved	This field is reserved.
13 MSKWDG_EOA_C	Interrupt enable (WDG_EOA_C status bit) 0 Interrupt disabled 1 Enables the STSR1[WDG_EOA_C] status bit to generate an interrupt indication.
14 Reserved	This field is reserved.
15 MSKWDG_EOA_S	Interrupt enable (WDG_EOA_S status bit) 0 Interrupt disabled 1 Enables the STSR1[WDG_EOA_S] status bit to generate an interrupt indication.
16 MSKERR_C	Interrupt enable (ERR_C status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_C] status bit to generate an interrupt indication.
17 Reserved	This field is reserved.
18 MSKERR_S2	Interrupt enable (ERR_S2 status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_S2] status bit to generate an interrupt indication.
19 MSKERR_S1	Interrupt enable (ERR_S1 status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_S1] status bit to generate an interrupt indication.
20 MSKERR_S0	Interrupt enable (ERR_S0 status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_S0] status bit to generate an interrupt indication.
21–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 EN	Self testing channel enable. The TEST conversions are enabled. Enables the TEST channel only in CPU mode. This bit should be set before starting the normal conversion and should not be changed while conversion is ongoing. This bit should be reset only after end of conversion for the last self test channel has been received. 0 Disabled 1 Enabled
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_STCR2 field descriptions (continued)

Field	Description
27 FMA_WDSERR	Fault mapping for the Watchdog Sequence error. By default a failure on the watchdog sequence is mapped on the NCF fault line. 0 NCF mapping 1 CF mapping
28 FMA_WDTERR	Fault mapping for the Watchdog Timer error. By default a failure on the watchdog sequence is mapped on the NCF fault line. 0 NCF mapping 1 CF mapping
29 FMA_C	Fault mapping for self test algorithm C. By default a failure on the self test C algorithm is mapped on the CF fault line. 0 NCF mapping 1 CF mapping
30 Reserved	This field is reserved.
31 FMA_S	Fault mapping for the self test algorithm BGAP. By default a failure on the Supply algorithm is mapped on the CF fault line. 0 NCF mapping 1 CF mapping

36.4.27 Self Test Configuration Register 3 (ADC_STCR3)

This register defines ADC self testing capabilities in CPU mode or CTU mode.

Address: 0h base + 348h offset = 348h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						ALG		0			MSTEP				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

ADC_STCR3 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ALG	Algorithm scheduling. ONE-SHOT mode algorithm scheduling:

Table continues on the next page...

ADC_STCR3 field descriptions (continued)

Field	Description
	• 00: Algorithm S (single step=MSTEP) • 01: Reserved • 10: Algorithm C (single step=MSTEP) • 11: Algorithm S (default) For test/debug purposes. SCAN mode algorithm scheduling: • 00: Algorithm S • 01: Reserved • 10: Algorithm C • 11: Algorithm S + Algorithm C (default) The baud rate for the execution of the selected algorithm is defined by the STBRR register.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 MSTEP	One shot mode: Current step for Algorithm S or Algorithm C. • MSTEP = 0 to (NUM_SUPPLY_STEPS-1) for algorithm S • MSTEP = 0 to (NUM_C_STEPS-1) for algorithm C Scan mode: This field is not used in scan mode as always single step execution (interleaved mode) is performed. It should be programmed to zero in scan mode. **NOTE:** All unused valued are RESERVED and should not be used.

36.4.28 Self Test Baud Rate Register (ADC_STBRR)

Used to set the baud rate for the selected baud rate in SCAN mode as well as the watchdog timer value.

Address: 0h base + 34Ch offset = 34Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													WDT			0								BR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STBRR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 WDT	The watchdog timer value is used to monitor that the algorithm sequence is correctly executed within the safe time period. The self testing watchdog is enabled setting the STAWxR[WDTE] control bits. Default value is 10 ms. A fixed pre-scaler runs on the ADCdig clock (80 MHz). (Prescalar value is fixed at 1024). 000 0.1 ms ((0008h * Prescalar) cycles at 80 MHz) 001 0.5 ms ((0027h * Prescalar) cycles at 80 MHz)

Table continues on the next page...

ADC_STBRR field descriptions (continued)

Field	Description
	010 1 ms ((004Eh * Prescalar) cycles at 80 MHz) 011 2 ms ((009Ch * Prescalar) cycles at 80 MHz) 100 5 ms ((0187h * Prescalar) cycles at 80 MHz) 101 10 ms ((030Dh * Prescalar) cycles at 80 MHz) 110 20 ms ((061Ah * Prescalar) cycles at 80 MHz) 111 50 ms ((0F42h * Prescalar) cycles at 80 MHz)
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BR	Baud rate for the selected algorithm in SCAN mode (MCR[MODE] = 1). 00h Max scheduling rate (NOMINAL rate) ... FFh Min scheduling rate (NOMINAL rate scaled by 255) This field should be programmed when ST_EN is zero i.e. before enabling self test channel.

36.4.29 Self Test Status Register 1 (ADC_STSR1)

Address: 0h base + 350h offset = 350h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				WDSERR	0	WDTERR	OVERWR	ST_EOC	0				WDG_EOA_C	0	WDG_EOA_S
W					w1c		w1c	w1c	w1c					w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_C	0	ERR_S2	ERR_S1	ERR_S0	0	STEP_C				0					
W	w1c		w1c	w1c	w1c											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR1 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 WDSERR	Watchdog sequence error of the ADC sub-system (check for algorithm step sequence). The WDSERR status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDSERR] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the Critical type or Non-Critical type lines according to the STCR2[FMA_WDSERR] mapping. **NOTE:** This bit is set only if STAWxR[WDTE] = ‘1’. 0 No failure 1 Failure occurred
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_STSR1 field descriptions (continued)

Field	Description
6 WDTERR	Watchdog timer error of the ADC sub-system (algorithm check for completion within safe time). The WDTERR status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDTERR] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the Critical type or Non-Critical type lines according to the STCR2[FMA_WDTERR] mapping. 0 No failure 1 Failure occurred
7 OVERWR	Overwrite error. Used to notify when the STSR1[ERRx] bit is overwritten by a newer one. The OVERWR status bit is cleared writing 1. A 0 write is an invariant operation. The new error status is written or discarded according to the MCR[OWREN] bit value. To avoid OVERWR indication, the ERRx status bit must be cleared (via software).
8 ST_EOC	Self Test EOC Bit. This bit is set along with EOC bit when end_of_conversion signal is received from ADC analog for self test channel. The ST_EOC status bit is cleared by writing 1b. A 0b write is an invariant operation. It generates an interrupt if enabled by STCR2[MSKST_EOC]. (Also, IMR[MSKEOC] has to be enabled.)
9–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 WDG_EOA_C	This bit indicates that Algorithm C has been completed. This bit is generated after the last STEP of algorithm C is executed. The WDG_EOA_C status bit is cleared by writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDG_EOA_C] = 1). This bit is set only if STAW4R[WDTE] = 1. For CTU conversions, this bit is significant only for Burst mode of operation.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 WDG_EOA_S	This bit indicates that Algorithm S has been completed. This bit is generated after the last STEP of algorithm S is executed. The WDG_EOA_S status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDG_EOA_S] = 1). This bit is set only if STAW0R[WDTE] = ‘1’. For CTU conversions, this bit is significant only for Burst mode of operation.
16 ERR_C	Indicates an error on the self testing channel (algorithm C). The ERR_C status bit is cleared writing 1b. A 0b write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKERR_C] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_C bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical type or Non-Critical type lines are asserted according to the STCR2[FMAx] mapping. 0 No C-algorithm ERROR 1 C-algorithm ERROR occurred
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ERR_S2	Indicates an error on the self testing channel (algorithm SUPPLY, step2). The ERR_S2 status bit is cleared by writing a 1b. A 0b write is an invariant operation. It generates an interrupt (“ipi_int_er” line) if enabled (STCR2[MSKERR_S2] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_S2 bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical / Non-Critical lines are asserted according to the STCR2[FMAx] mapping. 0 No ERROR occurred on the sampled signal 1 ERROR occurred on the sampled signal

Table continues on the next page...

ADC_STSR1 field descriptions (continued)

Field	Description
19 ERR_S1	Indicates an error on the self testing channel (algorithm SUPPLY, step1). The ERR_S1 status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKERR_S1] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_S1 bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical / Non-Critical lines are asserted according to the STCR2[FMAx] mapping. 0 No VDD ERROR 1 VDD ERROR occurred
20 ERR_S0	Indicates an error on the self testing channel (algorithm SUPPLY, step0). The ERR_S0 status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKERR_S0] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_S0 bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical / Non-Critical type fault lines are asserted according to the STCR2[FMAx] mapping. 0 No VREF ERROR 1 VREF ERROR occurred
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–26 STEP_C	Step of the algorithm C when an ERR_C has occurred. 0.. (NUM_C_STEPS-1) => algorithm C
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

36.4.30 Self Test Status Register 2 (ADC_STSR2)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 354h offset = 354h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVFL	0			DATA1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR2 field descriptions

Field	Description
0 OVFL	Overflow Bit This bit is set when the divisor is zero. Also, the STSR1[ERR_S1] bit is set if overflow occurs.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_S1 has occurred. Algorithm S (step1) => fractional part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA0	Test channel converted data when the ERR_S1 has occurred. Algorithm S (step1) => integer part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP

36.4.31 Self Test Status Register 3 (ADC_STSR3)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 358h offset = 358h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA1												0				DATA0											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR3 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_S2 has occurred. Algorithm S (step2) => TEST channel data = VREF/VREF
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA0	Test channel converted data when the ERR_S0 has occurred. Algorithm S (step0) => TEST channel data = VREF/VBGAP

36.4.32 Self Test Status Register 4 (ADC_STSR4)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 35Ch offset = 35Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA1												0				0											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR4 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_C has occurred. Algorithm C => TEST channel data
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_STSR4 field descriptions (continued)

Field	Description
20–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

36.4.33 Self Test Data Register 1 (ADC_STDR1)

The ADC_STDR1 register contains the most current result data for conversions executed during ADC self test. Additionally, the register contains status bits to indicate whether result data has been read and whether existing conversion data has been overwritten by a newer result.

NOTE

1. For Algorithm S step 1, the result data contained in this register is not the final step 1 result. The step 1 data is the conversion of the analog supply (VDDA). This VDDA conversion result is divided by the Algorithm S step 0 conversion data and the result is written to the ADC_STDR2 register.
2. The least significant 3 bits of the analog supply (VDDA) conversion executed during Algorithm S step 1 processing are always '0'. The conversion performed in Algorithm S Step 1 is not on the full scale analog supply. Instead, the voltage sampled is VDDA / 8. The prescaling is done to avoid issues that can arise in the analog portion of the ADC when VREFH = VDDA = 3.3 V, or VDDA is slightly higher than VREFH. The conversion result is multiplied by 8 before being written to this register to scale the result to the expected level, which results in the 3 least significant bits having a value of '0'.

Address: 0h base + 370h offset = 370h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												VALID	OWERWR	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				TCDATA											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STDR1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Valid data. Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.
13 OWERWR	Overwrite data. Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the MCR[OWREN] bit value.
14–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 TCDATA	Test channel converted data. - Unsigned format for Supply Self test - 2's complement for Capacitive Self test

36.4.34 Self Test Data Register 2 (ADC_STDR2)

NOTE

If overflow error occurs for STEP1 of Algorithm-S, STDR2 is not written.

Address: 0h base + 374h offset = 374h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FDATA												VALID	OWERWR	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				IDATA											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STDR2 field descriptions

Field	Description
0–11 FDATA	Fractional part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP for the algorithm S.
12 VALID	Valid data. Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.
13 OWERWR	Overwrite data.

Table continues on the next page...

ADC_STDR2 field descriptions (continued)

Field	Description
	Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the MCR[OWREN] bit value.
14–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 IDATA	Integer part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP for the algorithm S.

36.4.35 Self Test Analog Watchdog Register 0 (ADC_STAW0R)

This register provides control values associated with the ADC's supply self test step 0, which samples bandgap voltage.

Address: 0h base + 380h offset = 380h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWDE	WDTE	0		THRH											
W			▨													
Reset	0	0	0	0	0	1	1	1	0	0	1	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRL											
W	▨															
Reset	0	0	0	0	0	1	0	0	1	1	0	0	0	1	0	1

ADC_STAW0R field descriptions

Field	Description
0 AWDE	Analog watchdog enable (related to the algorithm S (step 0)) Enables/disables the comparison of the conversion result from the ADC's supply self test step 0 to the thresholds contained in this register (THRH and THRL) 0 Disabled 1 Enabled
1 WDTE	Watchdog timer enable (related to the algorithm S) Enables/disables the watchdog timer monitoring function for all ADC supply self test steps. The watchdog timer verifies: • Correct sequence of the algorithm (step sequence) • Execution of the algorithm within the safe time period as defined by STBRR[WDT] As soon as the watchdog timer is enabled the algorithm execution must be detected within the safe time period. The watchdog timer is reset each time the algorithm restarts. This bit should be set only in scan mode.

Table continues on the next page...

ADC_STAW0R field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for Algorithm S step 0 If the analog watchdog is enabled, the STSR1[ERRx] status bit is set if STDR1[TCDATA]) > THRH. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for Algorithm S step 0 If the analog watchdog is enabled, the STSR1[ERRx] status bit is set if STDR1[TCDATA]) < THRL. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

36.4.36 Self Test Analog Watchdog Register 1A (ADC_STAW1AR)

This register contains values associated with the ADC's supply self test step 1, in which the analog supply (VDDA) is sampled and a fixed point division is performed, dividing the conversion result by the result from step 0 (bandgap voltage). The integer portion of the result is handled separately from the fractional part, i.e., they have separate data registers and threshold fields. Thresholds in this register are applied to the integer portion of the result. Thresholds in the ADC_STAW1BR register are applied to the fractional portion of the result.

Address: 0h base + 384h offset = 384h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWDE	0			THRH											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

ADC_STAW1AR field descriptions

Field	Description
0 AWDE	Analog watchdog enable related to the algorithm S (step1).
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value (integer part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value (integer part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

36.4.37 Self Test Analog Watchdog Register 1B (ADC_STAW1BR)

This register contains values associated with the ADC's supply self test step 1, in which the analog supply (VDDA) is sampled and a fixed point division is performed, dividing the conversion result by the result from step 0 (bandgap voltage). The integer portion of the result is handled separately from the fractional part, i.e., they have separate data registers and threshold fields. Thresholds in this register are applied to the fractional portion of the result. Thresholds in the ADC_STAW1AR register are applied to the integer portion of the result.

Address: 0h base + 388h offset = 388h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRH												0				THRL											
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	0

ADC_STAW1BR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value (fractional part) for test channel for algorithm S (step 1)(unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value (fractional part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

36.4.38 Self Test Analog Watchdog Register 2 (ADC_STAW2R)

This register contains control values associated with the ADC's supply self test step 2, in which the ADC's high reference voltage is sampled and converted. Note that there is no upper threshold value in this register since the conversion result should ideally be FFFh (12-bit) or 3FF (10-bit).

Address: 0h base + 38Ch offset = 38Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWDE	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRL											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	1

ADC_STAW2R field descriptions

Field	Description
0 AWDE	Analog watchdog enable related to the algorithm S (step2).
1–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for channel x (unsigned coding). If the analog watchdog is enabled, the STSR1[ERR_S2] status bit is set if STDR1[TCDATA] < THRL. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

36.4.39 Self Test Analog Watchdog Register 4 (ADC_STAW4R)

This register contains ADC self test configuration controls associated with the ADC's capacitive self test (also referred to as "Algorithm C") sequence, which samples voltages from the ADC's internal capacitor matrix. The watchdog enable fields (AWDE and WDTE) affect all Algorithm C steps; the threshold fields (THRH and THRL) fields are specific to Algorithm C Step 0.

Address: 0h base + 394h offset = 394h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWDE	WDTE	0		THRH											
W			▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRL											
W	▨	▨	▨	▨												
Reset	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0

ADC_STAW4R field descriptions

Field	Description
0 AWDE	Analog watchdog enable (related to the algorithm C) Enables/disables the analog watchdog monitoring function for all ADC capacitive self test steps. 0 Disabled 1 Enabled
1 WDTE	Watchdog timer enable (related to the algorithm C). Enables/disables the watchdog timer monitoring function for all ADC capacitive self test steps. The watchdog timer verifies: • Correct sequence of the algorithm (step sequence) • Execution of the algorithm within the safe time period as defined by STBRR[WDT] As soon as the watchdog timer is enabled, execution must be detected within the safe time period. The watchdog timer is reset each time the algorithm restarts. This bit should be set only in scan mode. 0 Disabled 1 Enabled
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for step 0 of C algorithm.

Table continues on the next page...

ADC_STAW4R field descriptions (continued)

Field	Description
	This value is read by the ADC as twos-complement, but should be written positive. If the watchdog is enabled (STAW4R[AWDE] = 1) and STDR1[TCDATA] > STAW4R[THRH], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for step 0 of C algorithm. This value is read by the ADC as twos-complement, and it should be the negative of STAW4R[THRH]. If the watchdog is enabled (STAW4R[AWDE] = 1) and STDR1[TCDATA] < STAW4R[THRL], then STSR1[ERR_C] is set. **NOTE:** The settings in this register are applicable only to step 0 of the ADC's capacitive self test. Refer to the ADC_STAW5R register for settings applicable to other capacitive self test steps. The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

36.4.40 Self Test Analog Watchdog Register 5 (ADC_STAW5R)

This register contains ADC self test threshold values associated with all capacitive self test (also referred to as "Algorithm C") steps other than Step 0, which is controlled by fields in the ADC_STAW4R register.

Note that the threshold values in this register are distinctly different than those in the ADC_STAW4R register. The threshold values in this register do not represent a voltage conversion value relative to the ADC high reference voltage, but instead represent the upper and lower thresholds for the absolute difference between the voltage sampled and converted in an Algorithm C step and the voltage sampled and converted in Algorithm C Step 0. Ideally, the difference will be zero.

Address: 0h base + 398h offset = 398h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRH												0				THRL											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0

ADC_STAW5R field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for step N of C algorithm (N = 1 to CS-1). This value is read by the ADC as twos-complement, but should be written positive. If the watchdog is enabled (STAW4R[AWDE] = 1) and (STDR1[TCDATA] (step-N) – STDR1[TCDATA] (step-0)) > STAW5R[THRH], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for step 0 of C algorithm. This value is read by the ADC as twos-complement, and it should be the negative of STAW5R[THRH]. If the watchdog is enabled (STAW4R[AWDE = 1]) and (STDR1[TCDATA] (step-N) – STDR1[TCDATA] (step-0)) < STAW5R[THRL], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

36.4.41 Calibration, BIST Control and status Register (ADC_CALBISTREG)

This register controls the Offset calculation, Calibration and BIST functionality with Averaging option. The averaging function has no effect in other conversion modes.

Address: 0h base + 3A0h offset = 3A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OPMODE			TSAMP		0			Reserved							
W						▨										
Reset	1	1	0	0	0	0	0	0	0	0	1	1	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C_T_BUSY	Reserved	Reserved						Reserved	NR_SMPL		AVG_EN	TEST_FAIL	0		TEST_EN
W	▨												w1c	▨		
Reset	0	0	0	0	0	1	1	0	1	1	1	1	0	0	0	0

ADC_CALBISTREG field descriptions

Field	Description
0–2 OPMODE	This field specifies the operating mode of the ADC (normal/high accuracy). The high accuracy mode comes at a cost of 4 additional ADC cycles (80 Mhz/40 Mhz). In order to keep the same total conversion time as normal mode, the sample time has to be reduced. 000 Reserved 001 12-bit operating mode (normal) 010 Reserved 011 Reserved

Table continues on the next page...

ADC_CALBISTREG field descriptions (continued)

Field	Description
	100 Reserved 101 Reserved 110 12-bit operating mode (high accuracy, recommended) 111 Reserved
3–4 TSAMP	Test Sample period in Calibration, BIST and Offset calculation process. 00 22 cycles of ADC clk 01 8 cycles of ADC clk 10 16 cycle of ADC clk 11 32 cycle of ADC clk
5–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
16 C_T_BUSY	This bit indicates that ADC is doing internal operations for BIST or Calibration. This is set when the TEST_EN bit is set. 0 ADC is ready for use 1 ADC is busy in calibration or test operation
17 Reserved	This field is reserved.
18–23 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
24 Reserved	This field is reserved. The default value must not change. Always write the default value to this field.
25–26 NR_SMPL	Number of Samples for averaging. **NOTE:** The 16 samples setting is used only when the application needs quicker calibration (reduced time) to trade off with accuracy. For the best accuracy, it is necessary to use 512 samples. The minimum recommended number of samples is 32 samples. 00 16 samples 01 32 samples 10 128 samples 11 512 samples
27 AVG_EN	Average Enable (for Calibration only). 0 Disable 1 Enable
28 TEST_FAIL	Test Fail. This bit indicates the status of Calibration output. This bit sets automatically if tests are aborted prematurely by any normal conversion or tests complete but one of them fails. This bit is cleared by writing 1 to it or on restarting of test by writing 1 to TEST_EN 0 Test passed 1 Test failed

Table continues on the next page...

ADC_CALBISTREG field descriptions (continued)

Field	Description
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TEST_EN	Enable the test. 0 Disable the test 1 Enable the test (self clearing) (once set it cannot be overwritten till it resets)

36.4.42 Offset and Gain User Register (ADC_OFSGNUSR)

This register contains the user configured offset and Gain values used by ADC for data processing and have all the bits valid (MSB through LSB).

OFFSET and Gain Calibration values can be either positive or negative, so they need to be in 2's complement format with MSB as sign bit.

Address: 0h base + 3A8h offset = 3A8h

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>R</td><td colspan="6">0</td><td colspan="10" rowspan="2">GAIN_USER</td><td colspan="8">0</td><td colspan="8" rowspan="2">OFFSET_USER</td></tr>
<tr><td>W</td><td colspan="6"></td><td colspan="8"></td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

ADC_OFSGNUSR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 GAIN_USER	Gain User (2's complement format)
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 OFFSET_USER	Offset User (2's complement format)

36.5 Functional description

The following sections detail the ADCD interfaces as well as the functionality of this module.

36.5.1 Conversion

After power-up or reset, the ADC is in power-down mode until the MCR[PWDN] field is written. There are some configurations, listed below, available only in power-down mode, that must be handled before exiting power down.

- Clock selection for SAR Controller (AD_clk): MCR [ADCLKSEL]

After the above configurations, ADC need to be taken out from power-down mode by writing '0' into MCR [PWDN], Conversions can be initiated now in the following modes:

- Normal
- Injected
- CTU Triggered

36.5.1.1 Normal conversion

The conversion mode generally used to convert channels is called normal conversion. Programming the available Normal Conversion Mask Register(s) (ADC_NCMR*n*) selects the channel(s). Each channel can be individually enabled by setting a bit in the corresponding field of the NCMR*n* register.

- NCMR0 serves *precision* channels:
  - 0 - 15 (ADC0)
  - 0 - 15 (ADC1)
  - 0 - 4, 10, 14 (ADC2)
  - 0 - 7, 10 - 14 (ADC3)

The mask register(s) must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends. Conversion always starts from the lowest channel number selected and sequentially goes to the highest number.

36.5.1.1.1 Starting a normal conversion

After programming the MCR and NCMR0*n* register(s), a normal conversion is started via *software* by setting the MCR[NSTART] bit.

To start a normal conversion via *external trigger*, enable external trigger by setting the (MCR[TRGEN] bit. A programmed event (rising/falling edge depending on MCR[EDGE] bit) on the normal trigger input starts the conversion.

Detection of an active edge defined by MCR[EDGE] bit (rising=1, falling=0) on the external trigger will start the normal conversion.

36.5.1.2 Normal conversion operating modes

Two operating modes are available for normal conversion:

- One-Shot mode
- Scan mode

The MCR[MODE] bit determines which mode is used during a normal conversion. The first phase of the conversion process involves sampling the analog channel. The next phase is the conversion phase, during which the sampled analog value is converted to digital as shown in the figure below.

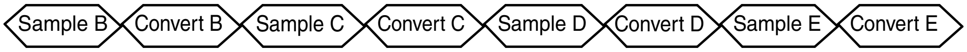


Figure 36-2. Normal conversion flow

36.5.1.2.1 One shot mode

In one shot mode (MCR[MODE] = 0) a sequential conversion specified in the following mask register(s) is performed only once:

- NCMR0

At the end of each conversion, the digital result of the conversion is stored into the corresponding data register(CDRx, x = channel number).

For example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in One-Shot mode. Conversion starts with channel B, followed by the conversion of channels D and E. At the end of conversion of channel E, the scanning of channels will stop.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time, MCR[NSTART]is reset by hardware, allowing the software to program a new start of conversion in advance. In this case, the newly requested conversion starts after completing the running/current conversion. However, for the sake of simplicity and straightforwardness, it is advisable to program the NSTART bit for the new conversion only after the completion of the running conversion. To know the end of conversion of running chain, application will either wait for an End of Chain (ECH) Interrupt or check the status of MSR[NSTART] prior to setting MCR[NSTART] again.

If the conversion chain is started by the external trigger, the trigger input will not be observed by the ADC block until the current conversion chain is finished. After the chain is over, the next chain can be invoked/triggered by another hardware trigger edge.

36.5.1.2.2 Scan mode

In scan mode (MCR[MODE] = 1), a sequential conversion of *n* channels specified in the NCMR*n* register(s) is continuously performed. At the end of each conversion, the converted result is stored into the corresponding data registers, as in One-Shot mode.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One-Shot mode, the MCR[NSTART] bit is not reset automatically in Scan mode. It can be reset by software when you need to stop Scan mode. In that case, the ADC completes the current chain and, after the last conversion, resets MSR[NSTART].

For example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in Scan mode. MCR[MODE] = 1 is set for Scan mode. Conversion starts from channel B, followed by the conversion of channels D and E. At the end of the conversion of channel E, the conversion of channel B starts again, followed by the conversion of channels D and E. This sequence repeats itself until MCR[NSTART] is reset by software.

This scan chain can be stopped by writing 0 to MCR[NSTART]. The conversion stops when the ongoing chain is finished.

NOTE

If the conversion is started by an external trigger, MCR[NSTART] is not set. As a consequence, once started, the only way to stop a Scan mode conversion started by an external trigger is to set the MCR[MODE] bit to 0.

36.5.1.2.3 End of conversion

In both modes, at the end of each conversion an end-of-conversion (EOC) interrupt is issued if enabled by the corresponding mask bit in the following register(s):

- CIMR0

Additionally, the EOC interrupt must enabled in the IMR register. After the conversion of all the channels selected in the NCMR*n* register(s) is completed and the conversion operation is considered finished, ISR[ECH] is set and an end-of-chain (ECH) interrupt is issued (if enabled in IMR[MSKECH]). The corresponding channel bit within the appropriate CEOCFR*n* register is updated to indicate that data is available on the data register (CDR*n*) of the respective channel. If there is no channel selected in the NCMR*n* register(s) and there is a start of conversion trigger, then ISR[ECH] is set and an ECH interrupt is immediately issued (if enabled).

NOTE

To add or remove channels to/from the ongoing chain:

1. Clear the MCR[NSTART] bit
2. Set/clear mask bit for the needed channels in NCMR*n*
3. Set the MCR[NSTART] bit

36.5.1.3 Injected conversion

An injected conversion is a conversion chain that can be injected at any time, including into an ongoing normal conversion chain. To do so, configure the injected mask register(s):

- JCMR0

As in a normal conversion, each internal or external channel can be individually selected for injected conversion.

- JCMR0 controls *precision* channels:
  - 0 - 15 (ADC0)
  - 0 - 15 (ADC1)
  - 0 - 4, 10, 14 (ADC2)
  - 0 - 7, 10 - 14 (ADC3)

Injected conversion can only be run in One-Shot mode, and can only interrupt a normal conversion. Please note, that a CTU conversion cannot be interrupted by an injected conversion. In this case, the injected conversion is discarded. When an injected chain is inserted into an ongoing normal chain, the ongoing normal channel is suspended and the injected channel request is processed. After completing the last channel in the injected chain, normal conversion resumes from the channel at which it was suspended as shown in the following figure.

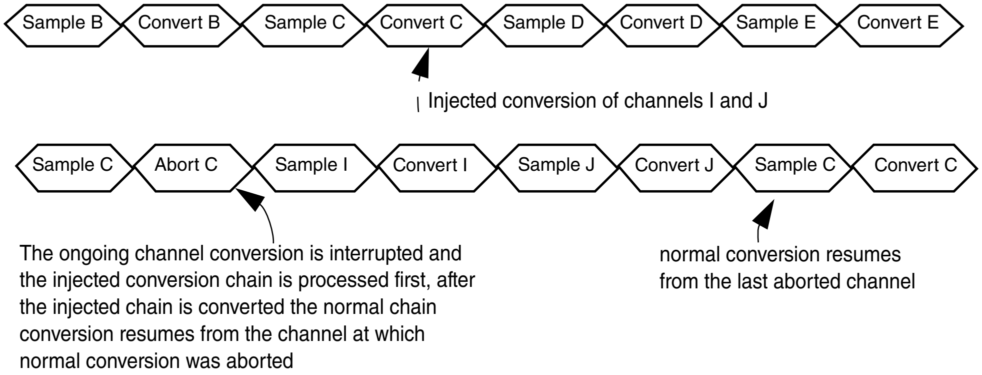

Sample B
Convert B
Sample C
Convert C
Sample D
Convert D
Sample E
Convert E
Injected conversion of channels I and J
Sample C
Abort C
Sample I
Convert I
Sample J
Convert J
Sample C
Convert C
The ongoing channel conversion is interrupted and the injected conversion chain is processed first, after the injected chain is converted the normal chain conversion resumes from the channel at which normal conversion was aborted
normal conversion resumes from the last aborted channel

Figure 36-3. Injected sample/conversion sequence

36.5.1.3.1 Starting injected conversion

The injected conversion can be started via *software* by setting the MCR[JSTART] bit to start an injected conversion; the current normal conversion is suspended and the injected chain is converted.

To start an injected conversion via *external trigger*, enable external triggering by setting MCR[JTRGEN]=1. A programmed event (rising/falling edge depending on MCR[JEDGE] bit) on the injection trigger input starts the injected conversion.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time, the MCR[JSTART] bit is reset, allowing software to program a new start of conversion in advance. In that case, the new requested conversion starts after the running conversion is completed.

At the end of each conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit in IMR) and at the end of the chain an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit in IMR). The status bit ISR[JECH] is also set.

If the content of all the injected conversion mask registers (JCMRn) is zero (i.e., no channel is selected) the interrupt JECH is immediately issued to resume normal operation or conversion mode.

The corresponding channel bit in the appropriate CEOCFR*n* register is updated to indicate that data is available in the data register (CDR) of the respective channel.

Once started, injected chain conversion cannot be interrupted by Normal or another Injected conversion. A CTU trigger has the higher priority; injected conversion has a lower priority. Therefore, only a CTU trigger can interrupt an ongoing injected conversion. In this case, the injected conversion will be discarded.

36.5.1.4 Aborting conversions

Two different abort functions are provided:

- A single channel abort
- A chain abort

You can abort the ongoing conversion by setting the MCR[ABORT] bit. The current conversion is aborted and the conversion of the next channel of the chain is immediately started. The abort action may take 1 to 4 cycles of the bus clock to abort the current conversion depending on the state of the conversion. At the start of any conversion and at the end of a particular conversion, when internal state counters are changing, then abort is delayed by a maximum of 3 cycles to take all states in stable condition. Abort also should not be programmed along with and within 3 cycles of programming the MCR[NSTART] or MCR[JSTART]. This may not have a significant impact from software point of view since on any SoC the next write access comes only after a minimum of 3 clock cycles.

NOTE

Programming Abort or Abort chain with NSTART or JSTART in idle condition has no effect.

For a Normal conversion chain having only one channel:

1. Abort re-runs the conversion again from the Abort point

If both Normal and Injected conversions are started by hardware trigger and they are coming in a particular cycle-dependent manner, e.g.,

1. When Abort is programmed one cycle after start of Injected conversion followed by a start of normal conversion in the next cycle of Abort, or
2. When Abort comes one cycle after the start of a normal conversion followed by the start of an Injected conversion in next cycle,

then Abort may or may not be detected and the Missing of Abort will consume maximum of 1 μs (one conversion time) more time for the execution of to-be-aborted conversion.

In the case of an abort operation, the MSR[NSTART/JSTART] bit remains set, if not in last channel of chain, and the MCR[ABORT] bit is reset as soon as the channel is aborted. The EOC corresponding to the aborted channel is not generated. This behavior is true for Normal or Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported by generating an ECH interrupt.

It is also possible to abort the current chain of conversions by setting the MCR[ABORTCHAIN] bit. In that case, the behavior of the ADC depends on the MCR[MODE] bit (One-Shot/Scan) for Normal conversion. In One-Shot mode, MSR[NSTART] is automatically reset together with the MCR[ABORTCHAIN]. Otherwise, in Scan mode, a new chain is started, and the current chain is aborted. The EOC of the current aborted conversion is not generated, but an ECH interrupt is generated to signal the end of the chain. For Injected conversion, the behavior of ABORTCHAIN is same as for One-shot Normal conversion. In this case, MSR[JSTART] is automatically reset together with the MCR[ABORTCHAIN].

When an ABORTCHAIN is requested while an injected chain is running over a suspended normal chain, both injected and normal chains are aborted; both MSR[NSTART] and MSR[JSTART] are reset. ABORT and ABORTCHAIN requests are ignored during CTU conversion. ABORT requests are also discarded during Self-Test conversion.

36.5.2 Crosstriggering Unit interface

The Crosstriggering Unit (CTU) interface enhances the injected conversion capability of the ADC. The CTU contains multiple event inputs that can be used to select the channels to be converted from the appropriate event configuration register. The CTU/ADC interface is shown in the figure below. The CTU generates a trigger and a channel number to be converted. A single channel is converted for each request. After performing the conversion, the ADC returns the result.

The conversion result is also saved in the corresponding data register and it is compared with watchdog thresholds if requested.DMA and interrupt feature of ADC can also be used during CTU conversion.

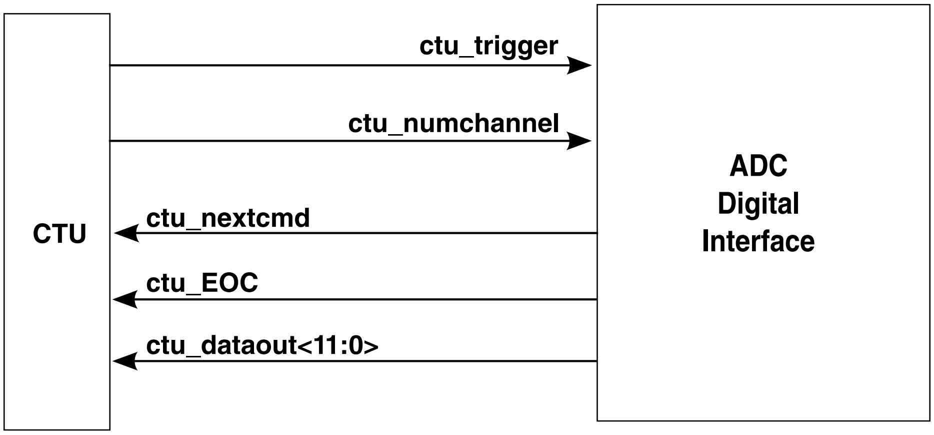


Figure 36-4. ADC Crosstriggering Unit

The CTU interface has one mode of operation, Control.

The CTU interface is enabled by programming the MCR[CTUEN] bit.

36.5.2.1 CTU Control mode

In CTU Control mode, if enabled via MCR[CTUEN], the CPU is able to write in the ADC registers, but it cannot start a conversion. Conversion requests can be generated only by a CTU trigger. If a normal or injected conversion is requested when the CTU is enabled (MCR[CTUEN]), it will be automatically discarded.

Along with CTU trigger the ctu_numchannel is taken as channel and CTU conversion starts. MSR[CTUSTART]is set automatically at this point and it remains set unless the CTU is disabled by resetting MCR[CTUEN].

CTU conversions must be requested when calibration is over. If a CTU trigger is received during calibration, the calibration stops immediately in order to satisfy the CTU request. Calibration will fail in this case.

If another CTU trigger is received before the ctu_nextcmd signal, the new request is discarded.

36.5.3 ADC clock prescaler and sample time settings

The clock (AD_clk) provided to the ADC's SAR controller, which controls the ADCA, must satisfy particular conditions of frequency and duty cycle. The maximum acceptable frequency is 80 MHz with a duty cycle equal to 50% (±5%).

The AD_clk frequency can be scaled by programming the MCR[ADCLKSEL] bit. If this bit is set, AD_clk frequency is the same as the bus clock. Otherwise, AD_clk frequency is half of the bus clock. MCR[ADCLKSEL] can only be written in power-down, when MCR[PWDN] = 1.

Sampling times are controlled by settings in the Conversion Timing Register (CTR*n*).

- CTR0 controls sampling time for *precision* channels:
  - 0 - 15 (ADC0)
  - 0 - 15 (ADC1)
  - 0 - 4, 10, 14 (ADC2)
  - 0 - 7, 10 - 14 (ADC3)

The exception is the Temperature Sensor channel, which always uses the CTR1 value. See Conversion Timing Register 1 (ADC_CTR1) for details.

36.5.4 Presampling

Presampling allows the ADC internal capacitor to precharge or discharge to a defined level before it starts the actual sampling/conversion of the analog input coming from the pads. This is useful for resetting information (history effect/offset) regarding the last converted data. During presampling, the ADC samples the internally generated voltage while, during sampling, the ADC samples the analog input coming from the pads.

Presampling can be enabled/disabled on an individual channel by setting the corresponding bit in the applicable PSR*n* register.

- PSR0 controls precision channels

After enabling the presampling for a channel, the normal sequence of operation is presampling + sampling + evaluation for that channel. Sampling of the channel can be bypassed by setting PSCR[PRECONV]. When sampling of a channel is bypassed, the presampled voltage will be converted.

See Conversion time for the conversion's timing equation.

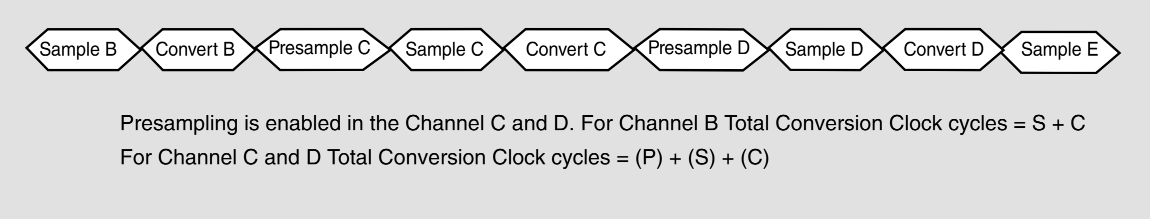

Sample B
Convert B
Presample C
Sample C
Convert C
Presample D
Sample D
Convert D
Sample E
Presampling is enabled in the Channel C and D. For Channel B Total Conversion Clock cycles = S + C
For Channel C and D Total Conversion Clock cycles = (P) + (S) + (C)

Figure 36-5. Presampling sequence

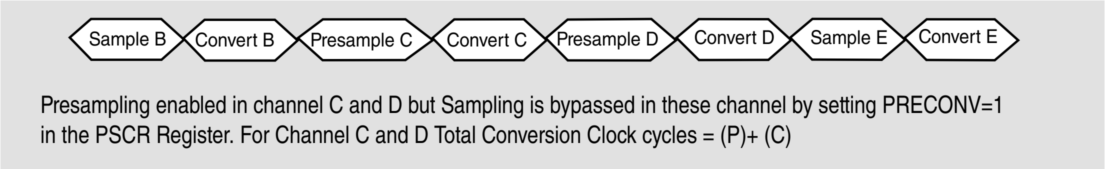

Sample B
Convert B
Presample C
Convert C
Presample D
Convert D
Sample E
Convert E
Presampling enabled in channel C and D but Sampling is bypassed in these channel by setting PRECONV=1
in the PSCR Register. For Channel C and D Total Conversion Clock cycles = (P)+ (C)

Figure 36-6. Presampling sequence with PRECONV = 1

36.5.4.1 Presampling channel enable

The presampling channels are selected by programming the presampling register (Presampling Control Register (ADC_PSCR)). These are used to enable analog switches that samples an internally generated voltage. It is possible to select presample voltage from four internally available voltages depending on the value of PREVAL*x* [*x* = 0,1,2] fields in the PSCR register as given in Table 36-1.

Table 36-1. Presample voltage selection

PREVAL*n*	Presampling voltage
00b	Presample voltage 1: VSS_HV_ADV (High voltage ground for ADC)
01b	Presample voltage 2: VDD_HV_ADV/8 (Power High voltage Supply for ADC, SGEN divided by 8)
10b	Presample voltage 3: VSS_HV_ADRE1 (High voltage ground for ADC/TSENS)
11b	Presample voltage 4: VDD_HV_ADRE1 (High voltage reference for ADC/TSENS)

The PREVALn fields are associated with specific channels, as shown in Table 36-2.

Table 36-2. PREVAL*n* channel association

Field	Channels
PREVAL0	• 0 - 15 (ADC0) • 0 - 15 (ADC1) • 0 - 4, 10, 14 (ADC2) • 0 - 7, 10 - 14 (ADC3)

The temperature sensor channel is an exception. It can be mapped with any of the channels, but it uses the PREVAL1 value to select the presample voltage because its physical properties may be different than the other internal precision channels.

36.5.5 Programmable analog watchdog

NOTE

> The number of analog watchdogs and channels may vary depending on the ADC instance on your chip. Please see the chip-specific section for how many are available on each instance.

The analog watchdogs are used for determining whether the result of a conversion lies within a given guard area (as shown in Figure 36-7) specified by an upper and a lower threshold value named THRH and THRL, respectively.

After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside the guard area, then the corresponding threshold violation interrupt is generated.

The comparison result is stored as WTISR[WDGxH] and WTISR[WDGxL], as explained in the following table. Depending on the WTIMR[MSKWDGxL] and WTIMR[MSKWDGxH] mask bits, an interrupt is generated upon threshold violation.

Table 36-3. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRH ≤ converted data ≤ THRL

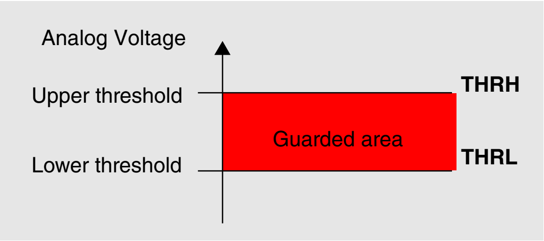


Figure 36-7. Guarded Area

36.5.5.1 Analog watchdog setting

For each watchdog there is one Threshold value (Low and High) Register (THRHLR).

The analog watchdog for each channel can be enabled independently by programming CWENR*n* register bits.

- CWENR0 controls *precision* channels:
    - 0 - 15 (ADC0)
    - 0 - 15 (ADC1)
    - 0 - 4, 10, 14 (ADC2)
    - 0 - 7, 10 - 14 (ADC3)

The availability of these registers depends on device configuration.

Threshold values for each channel can be selected independently from a maximum of 16 threshold registers (THRHLRx,(x=0..15)) by the WSEL_CHx field of CWSELRx (x=0..11) register.

Each CWSELRx register holds selection for 8 consecutive channels. CWSELR0 holds 8 WSEL_CH fields for channel 0 to 7. Likewise CWSELR1 is for channel 8 to 15 and so on. The availability of fields and registers depends on device configuration.

If the converted value for a particular channel lies outside the range specified by threshold values, then the corresponding bit is set in the Analog Watchdog Out of Range Register (AWORR).

For example, if channel number 12 is to be monitored with the threshold values in register THRHLR3, then CWSELR1[WSEL_CH4] needs to be programmed with 3. The enabling will be done by setting the bit corresponding to channel 12 in the CWENR0 register.

A set of threshold values (THRHLRx) can be linked to several ADC channels. The threshold values to be selected for a channel needs be programmed only once in the CWSELRx register.

36.5.6 DMA functionality

A Direct Memory Access (DMA) request can be programmed after the conversion of every channel by setting the respective masking bit in the following register(s):

- DMAR0

The DMA masking registers must be programmed before starting any conversion.

The DMA transfers can be enabled using the ADC_DMAE[DMAEN] bit. When the DMAE[DCLR] bit is set, the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

36.5.7 Interrupts

ADC generates the following maskable interrupt signals:

- End-of-conversion interrupt (EOC) request
- End-of-chain interrupt (ECH) request
- End-of-injected conversion interrupt (JEOC) request
- End-of-injected chain interrupt (JECH) request
- End-of-CTU conversion interrupt (EOCTU) request
- Watchdog threshold interrupt (WDGxL and WDGxH) requests

Interrupts are generated during the conversion process to signal events, like the end of a conversion, as explained in the register description for the Interrupt Status Register (ISR). The ISR and the Interrupt Mask Register (IMR) allow you to check and enable the interrupt request.

Interrupts can be individually enabled on a channel-by-channel basis by programming the Channel Interrupt Mask Register (CIMR).

A Channel Pending Register (CEOCFR*n*) is also provided in order to signal which of the channels' measurement has been completed.

Analog Watchdog interrupts are enabled by Watchdog Threshold Interrupt Mask Register (WTIMR) and the status can be checked at Watchdog Threshold Interrupt Status Register (WTISR). The watchdog interrupt source sets two fields, WDGxH and WDGxL, for each channel monitored in the WTISR.

In particular, the EOC, ECH, EOCTU, JEOC, and JECH interrupt lines are combined (OR-ed) on the same line. The EOC interrupt request is cleared by clearing either ISR[EOC] or all bits of CEOCFR. The JEOC interrupt request is cleared by clearing either ISR[JEOC] or all bits of CEOCFR. The EOCTU interrupt request is cleared by clearing either ISR[EOCTU] or all bits of CEOCFR. Also, the self-test end of algorithm interrupts STSR1[WDG_EOA_S] and STSR1[WDG_EOA_C] are combined.

All WDGxH and WDGxL requests are combined on the second line.

Another interrupt on the third line is a combination of WDSERR, WDTERR, ERR_S0, ERR_S1, ERR_S2, and ERR_C in STSR1.

The ISR register contains the interrupt pending request status. If you want to clear a particular interrupt event status, then writing a 1 to the corresponding status bit will clear the pending interrupt flag. (At this write operation, all the other bits of the ISR register must be programmed to 0.)

36.5.8 Power-Down mode

The analog part of the ADC can be put in Power-Down mode by setting the MCR[PWDN] bit, which shuts off the ADCA and stops the clock in the SAR controller. After power-on or device reset, the ADCA is kept in Power-Down mode by default, so this state must be exited before starting any operation by resetting the MCR[PWDN] bit. In Power-Down mode, no conversion and calibration can be started. Trigger from CTU during power down is discarded. In the ADC power-down mode, the CTU must not start any conversion.

The MCR[PWDN] bit may be programmed high at any time. However, if a conversion is ongoing, then the ADCA cannot be sent immediately into Power-down mode. In fact, the ADC enters Power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN function.

The ADC status in MSR[ADCSTATUS] field shows Power-down status only when ADC enters in Power-down mode.

If CTU is enabled and MSR[CTUSTART] is 1, then the MCR[PWDN] bit cannot be set. When CTU Control mode enabled, the application needs to reset the CTUEN bit before entering Power-Down mode.

After the power-down phase is complete, the process ongoing before the power-down phase must be restarted manually .

NOTE

Resetting the MCR[PWDN] bit and setting the MCR[NSTART] or MCR[JSTART] bit during the same cycle is forbidden.

36.5.9 Auto-Clock-Off mode

To reduce power consumption in Idle mode (without going into Power-Down mode), an auto-clock-off feature can be enabled by setting MCR[ACKO]. When enabled, the analog clock is automatically switched off when no operation is ongoing (that is, when no conversion has been programmed).

NOTE

This feature must remain off during calibration.

36.5.10 Calibration and Self Test

The ADC is calibrated and tested runtime through the same set of test conversions. The test result is used for computation and stored during the calibration process and only checked in self test. The two modes of running tests are as follows:

1. High-Accuracy mode/Calibration. Needs to be run after power-on reset and whenever required in runtime operation. Calibrates the ADC and updates calibration-related registers.
2. Quick-Check mode/Self-Test. Used for runtime functional self-tests to improve reliability. Catches any runtime fault or accuracy shift from previously calibrated values. Please see the chip-specific configuration information as self-test may not be available in all ADC instances.

The following sections describe both and the steps for executing them.

36.5.10.1 Calibration (High Accuracy mode)

The functioning of the analog block of ADC depends on the accuracy of values of its various parts and parameters, such as the value of capacitors, the gain of the amplifier, and so on. Due to variations in manufacturing and various runtime environmental effects, the actual values may differ from the ideal values for which the ADC is targeted.

Because of these deviations, the ADC's converted values contain errors. To eliminate these errors, the ADC must be calibrated prior to any conversion.

In the calibration process, a fixed known reference voltage (VrefH) is sampled many times and converted under various controlled conditions to check for deviations between these converted values and predefined values.

The deviations, known as offsets or modified values, are used to eliminate errors during the data processing of normal conversions.

Calibration needs to be performed after every power-up/destructive reset and whenever required in runtime operation. The applicability of destructive reset depends on device configuration. Please check the ADC configuration for the device.

Note

> The ADC must be recalibrated if the operating conditions (particularly VrefH) change significantly. Calibration should also be performed again if the Self-Test indicates that a recalibration is needed.

Recommended settings:

- Clock frequency = 40 MHz
- Averging – Maximum (512)

The calibration process will take ~12ms of time with the above recommended settings.

NOTE

> If the ADC clock frequency is > 40MHz during calibration, the accuracy of the calibration process will be compromised, hence the performance of the ADC. Thus, it is strongly recommended to run the calibration with 40 MHz clock.

As calibration is a time consuming process, it can be performed after a long interval – probably several hours of operation – but it is not necessary to recalibrate frequently. The goal is to maximize the possible accuracy delivered from conversion and to catch any structural fault to flag the error.

The calibration routine sets the fail flag (CALBISTREG[TEST_FAIL]) to indicate:

- A fault in ADC
- A premature abort of the calibration

Calibration is aborted if a normal conversion is initiated during calibration; this is called *premature termination*. The terminating conversion and the immediate next one to it will have invalid result in this case.

CALBISTREG[TEST_FAIL] = 1 indicates that the ADC health is not good and that it will not meet specifications.

To execute this routine, follow these steps:

1. Configure the ADC to 40 MHz.
    - Put the ADC in power down mode: set MCR[PWDN]
    - Configure clock divider selection: reset MCR[ADCLKSEL]
    - Power-up the ADC: reset MCR[PWDN]
2. Configure the Calibration, BIST Control, and Status Register (CALBISTREG). The default values are set for maximum accuracy (recommended).
3. Start calibration: set CALBISTREG[TEST_EN].

4. Wait for calibration to complete. Check MSR[ADCSTATUS] or till CALBISTREG[C_T_BUSY] becomes 0.
5. Check the value of MSR[CALIBRTD] to determine whether calibration was successful.
6. If MSR[CALIBRTD] = 1, calibration was successful; otherwise, it failed, and CALBISTREG[TEST_FAIL] = 1.
7. If required, reconfigure the ADC clock frequency for normal operation.

36.5.10.2 Self test (Quick-Check/Safety mode)

For devices used for very critical applications requiring high reliability it is important to check at regular intervals if the ADC is functioning correctly. For this purpose, Self Testing feature (Quick Check) has been incorporated inside ADC.

The following testing algorithms are available for checking:

- Supply Self test: algorithm S. It includes the conversion of the bandgap, supply and VREF voltages. It includes a sequence of three test conversions (steps). The supply test conversions must be an atomic operation (no functional conversions interleaved).
- Capacitive Self test: algorithm C. It includes a sequence of 12 test conversions (steps) to check the capacitive array.

The capacitive test of ADC requires only 12 steps to test itself.

The capacitive array is used as DA converter to generate from V_{REFH} 12 different voltages by distributing and partially discharging portions of the capacity array. These 12 voltages are sampled using some redundant capacities. C algorithm may be compared to providing 12 different voltages to ADC and validate the conversion result.

Two kinds of tests are performed in this mode, as shown in the following table.

Table 36-4. Quick-Check mode tests

STCR3 [ALG]	STCR3 [MSTEP]	Description	Outcome	Comment
00 supply test	0	Supply self-test (band gap voltage)	Measures the internal band gap voltage	—
	1	Supply self-test (analog supply)	Measures the analog supply voltage (VDDA) and divides the result by the bandgap voltage (result from Algorithm S step 0). The VDDA conversion result is written to ADC_STDR1[TCDATA]. The final result (after division) consists of an integer portion and a fractional portion, both of which	The least significant 3 bits of the analog supply (VDDA) conversion executed during Algorithm S step 1 processing and written to ADC_STDR1[TCDATA]) are always '0'. The conversion performed in Algorithm S Step 1 is not the full scale analog supply. Instead, the voltage sampled is VDDA / 8. The prescaling is done

Table continues on the next page...

Table 36-4. Quick-Check mode tests (continued)

STCR3 [ALG]	STCR3 [MSTEP]	Description	Outcome	Comment
			are written to the ADC_STDR2 register.	to avoid issues that can arise in the analog portion of the ADC when VREFH = VDDA = 3.3 V, or VDDA is slightly higher than VREFH. The conversion result is multiplied by 8 within the digital part of the ADC to scale the result to the expected level, which results in the 3 least significant bits having a value of '0'.
	2	Supply self-test (reference voltage high)	Measures the high reference voltage of the ADC	—
01	Reserved	—	—	—
10 capacitive test	0–11	One of the ADC-BIST steps is being run	The difference or error of individual value from the previously calibrated value is being returned (unsigned) as an ADC result that is compared with a programmed value in the analog watchdog register(s) to detect any fault. The difference or error may be caused by any runtime fault, runtime accuracy shift, or device noise.	Ideally, the result value is expected to be 0. If the returned value is higher, this indicates runtime fault or accuracy shift. This indicates that ADC should be recalibrated in High-Accuracy mode to check whether the reported error can be handled by recalibrating or whether permanent damage to ADC has occurred.
11	Supply test followed by Capacitive test in same sequence as shown above	—	—	—

This mode is described in the following sections.

36.5.10.2.1 Self test initialization and application information

ADC samples the internal V_{REFH} for the C algorithm and samples different supplies (bandgap, V_{DD} and V_{REFH}) for the supply algorithm. It also provides signals for the following:

- Scheduling self-testing algorithms using configuration registers
- Monitoring the converted data using analog watchdog registers
- Flagging the error to FCCU in case a failure occurs in any of the algorithms

ADC mode	TEST channel (ADCdig)	TEST channel (CTU)
CPU mode (MCR[CTUEN] = 0)	Yes	No

Table continues on the next page...

ADC mode	TEST channel (ADCdig)	TEST channel (CTU)
	• One-Shot mode • Scan mode	
CTU Control mode	No	Yes

36.5.10.2.2 CPU mode

In this case, the test channel works similarly to a normal conversion. The test channel is enabled by setting STCR2[EN].

Self-test conversions are executed by interleaving the Normal conversions. The sequencing of steps of the selected algorithm for test channel depends on the operating mode of normal conversion, selected by MCR[MODE].

In One-Shot mode, if the test channel is enabled, only one step of the selected self-testing algorithm is executed at the end of the chain. The step number and algorithm to be executed are programmed in the STCR3 register. So, in One-Shot mode, the sequence is as follows:

1. Program NCMR*n* register(s) to select channels to be converted for normal conversion.
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (for 1 MSPS operation)
3. Program MCR[MODE] = 0 to select One-Shot mode.
4. Program sampling duration for self test conversion in STCR1[INPSAMPx].
5. Select the self-testing algorithm in STCR3[ALG]. The default is algorithm S.
6. Enable the self-testing channel by setting STCR2[EN].
7. Start the normal conversion by setting MCR[NSTART].
8. All normal conversions are executed as usual.
9. At the end of allnormal conversions, the step number programmed in STCR3[MSTEP] of the self-testing algorithm selected by STCR3[ALG] is executed similar to a normal functional channel.
10. On receiving an end-of-conversion for the test channel, the digital result is written in STDR1[TCDATA] andSTDR1[VALID] is set. Also, ISR[EOC], ISR[ECH], and STSR1[ST_EOC] are set. See the register descriptions (Self Test Data Register 1 (ADC_STDR1), Interrupt Status Register (ADC_ISR), and Self Test Status Register 1 (ADC_STSR1)) for details of settings for each test.
11. The state machine returns to the Idle state.

For example: in a device with channels A-B-C-D-E-F-G-H, channels B-D-E are to be converted in One-Shot mode. MODE = 0 is set for One-Shot mode. Refer to Conversion. Conversion for channels B-D-E are done. After channel E test channel conversion is done and ISR[ECH] and ISR[EOC] are set.

MSR[NSTART] is automatically set when the normal conversion starts; it is reset at the end of conversion for the test channel.

In Scan mode, the consecutive steps of the selected self test algorithm are converted continuously at the end of each chain of normal conversions. The number of channels converted at the end of each chain is 1 (except for algorithm S, in which all the steps are performed at once without any functional conversion interleaved). So, in Scan mode the sequence is as follows:

1. Program the NCMR*n* registers to select channels to be converted for normal conversion.
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (for 1 MSPS operation)
3. Program MCR[MODE] = 1 to select Scan mode.
4. Program sampling duration for self test conversion in STCR1[INPSAMPx].
5. Select the self-testing algorithm in STCR3[ALG]. By default, all algorithms are selected (that is, all algorithms are executed step-by-step one after the other).
6. Enable the self-testing channel by setting STCR2[EN].
7. Start the normal conversion by setting MCR[NSTART].
8. All normal conversions are executed as usual.
9. At the end of chain forthe normal conversions (assuming a default value of STCR3[ALG]), all steps of algorithm S are performed (as algorithm S is always atomic). MSR[SELF_TEST_S] is set.
10. At the end of conversion of test channel for the last step of algorithm S, the digital result is written in STDR1[TCDATA] and STDR1[VALID] is set. At the same time, MSR[SELF_TEST_S] is reset. For step 1, the integral part and fractional part are written in STDR2[IDATA] and STDR2[FDATA]. Also, ISR[EOC], ISR[ECH], and STSR2[ST_EOC] are set.
11. The next normal conversion chain starts.
12. At the end of the normal conversion chain, Step0 of the C algorithm is executed.
13. At the end of conversion of test channel for Step0 of the C algorithm, the digital result is stored in STDR1[TCDATA] and STDR1[VALID] is set (if MCR[OVERWR] is set). Also, ISR[EOC], ISR[ECH], andSTSR2[ST_EOC] are set.
14. The next normal conversion chain starts.
15. At the end of the normal conversion chain, step1 of C algorithm is executed.
16. This process continues for all steps of all algorithms.
17. The state machine returns to the Idle state when MCR[NSTART] is written to 0.

NOTE

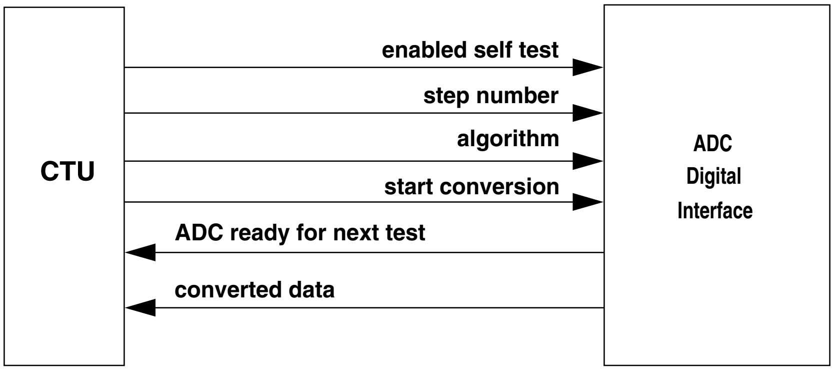
- Instead of starting a normal conversion with software (by setting MCR[NSTART]), if it is started by external trigger, the test channel behavior remains the same.

- Test channel conversion is not performed during injected conversions. It is performed only during normal conversions.
- If, during a test channel conversion, injection conversion arrives, then the test channel is aborted (just as a normal functional channel) and injected conversions are done. After injected conversions are completed, the test channel resumes from the step at which it was aborted. In this case, MSR.SELF_TEST_S remains high during the injected conversion.
- For self testing, MCR[MODE]should be programmed at least one cycle *before* setting MCR[NSTART]. It should not be changed thereafter until a conversion is ongoing.

36.5.10.2.3 CTU mode

For this chip, the operating mode is CTU Control mode.

When MCR[CTUEN] is set, the test channel conversion can be started only by the CTU interface; software cannot start it. STCR2[EN] does not have any effect in CTU mode. The CTUEN bit should not be changed while normal conversion is ongoing. The interface between the CTU and ADCD (for self test) is shown here:



For self testing conversions in CTU mode, the CTU asserts an indication that enables self test along with the CTU trigger that starts the self test conversion. The algorithm and the step number to be executed is put on the interface respectively.

NOTE

As already mentioned for algorithm S, the three steps must be atomic. In CTU mode, this is managed by the CTU itself; that is, the CTU has to send three triggers (one for each step) asserting the indication that enables self test and updating the step number for each step.

36.5.10.2.4 Abort and Abort Chain for self-test channel

Setting MCR[ABORT] during self-test channel has no effect.

In One-Shot mode, if MCR[ABORTCHAIN] is set when the test channel conversion is ongoing, the test channel is aborted and ECH is set. In this case, EOC for the test channel is not generated.

In Scan mode, if MCR[ABORTCHAIN] is set when test channel STEP N is ongoing, then the test channel STEP N is aborted and the next chain conversion starts. At the end of this chain, STEP N conversion is performed again. (If algorithm S is ongoing, the full algorithm is executed again.)

36.5.10.2.5 Self-test analog watchdog

ADCD also provides a monitor for the values returned by the ADC analog macro for self-test algorithms. The analog watchdogs are used to determine whether the result of conversion for self-test algorithms lie in a particular guard area. For this purpose, separate Self-Test Analog Watchdog Registers are provided for each algorithm.

If the analog watchdog feature is enabled (STAWxR[AWDE] = 1), a comparison is performed between the converted value and the threshold values after the conversion of each step of an algorithm. If the converted value does not lie between the upper and lower threshold values specified by the Analog Watchdog Register of the particular algorithm, the corresponding error bit (STSR1[ERR_x]) is set and the step number during which the error occurred is updated in STSR1[STEP_x] (in case of the C algorithm) and erroneous data is written in STSR4[DATAx]. STSR1[ERR_x] generates an interrupt if enabled by the corresponding mask bit in STCR2. The fault indication is also given to the FCCU via the Critical and Non-Critical fault lines, so that necessary action can be taken at chip level.

The analog watchdog feature works differently in the case of algorithm S. As already mentioned, algorithm S is always an atomic operation. So, separate error bits are provided in STSR1 for each step of algorithm S to avoid an overwrite in case an error occurs in more than one step. Hence, there are separate mask bits for each step in STCR1. For the same reason, separate fields in the status registers (STSR2 and STSR3) exist to store erroneous data for each step.

In step 0 of supply algorithm, ADC measures the band gap voltage (+1.2V), which is assumed to be stable with very little variation (up to +/- 1.5%). The converted data of step 0 is measured with high (THRH) and low (THRL) threshold defined in STAW0R if enabled (STAW0R[AWDE]). The STSR1[ERR_S0] is set to '1' if threshold is violated; once set, it is cleared to '0' by writing '1' to it.

In step 1 of supply algorithm, ADC measures the Analog supply (Vdda) for the ADC, and then a fixed point division is performed with the result of step 1 (ADC supply voltage) and step 0 (band gap voltage). It takes around 26 cycles after EOC for step 1 to get a divided value, and it is the divided value on which analog watchdog checks are applied. So, the value to be compared for step 1 contains the integer as well as fractional part; two registers (STAW1AR and STAW1BR) are provided for this. The comparison is done first for the integer part using the threshold values programmed in STAW1AR. If the integer part lies in the range, the fractional part comparison is skipped; otherwise, it is compared with the values programmed in STAW1BR. The following table summarizes this feature for step 1.

Table 36-5. Algorithm S (step 1) threshold comparison

STDR2[IDATA] (integer part)	STDR2[FDATA] (fractional part)	STSR1[ERR_S1]
> STAW1AR[THRH]	Any value	Set
< STAW1AR[THRL]	Any value	Set
== STAW1AR[THRH]	> STAW1BR.THRH	Set
== STAW1AR[THRL]	< STAW1BR.THRL	Set

In step 2 of supply algorithm, ADC measures the High Reference voltage (Vrh); (VREF/VREF) is measured in order to check the integrity of sampling signal. For this particular conversion, no higher threshold value is required as the ideal value is 0xFFF. Only the lower threshold value is programmed in STAW2R.

In the supply self-test algorithm, the band gap is measured with regard to the reference voltage (Vrh) in step 0. The result is capable of catching mismatches between them. Step 1 measures the supply with regard to reference (Vrh), and the result is taken to do a ratio (Vdda/Vbg). If the band gap voltage remains within +/-1.25% of its ideal value of 1.2 V, then by setting appropriate values of THRH and THRL in watchdog registers, the ADC self-test can catch more than 4% variation in supply and reference faithfully.

Algorithm C has total 12 steps. Output data of step 0 is compared with the threshold values stored in STAW4R, if enabled in the same register. The result of step 0 is used as an offset for rest of the steps (step 1 to 11). The absolute difference between the converted data of each of the steps (1 to 11) with the data of step 0 are compared with the threshold values of STAW5R, if enabled in STAW4R. Error in any step is indicated by STSR1[ERR_C] and the data in error case is captured in STSR4.

Calculation of Threshold values:

S1 Algorithm: Calculated voltage comparison method:

The user software can implement a method of calculating S1 algorithm IDATA, FDATA, the THRH values, and the THRL values to voltages and then perform a comparison of the calculated voltage to the calculated THRH voltage and calculated THRL voltage.

The steps involved in this method are as follows:

1. Calculate the high threshold voltage:

   (STAW1AR [THRH] + (STAW1BR [THRH] / 4096)) * Vbandgap
2. Calculate the low threshold voltage:

   (STAW1AR [THRL] + (STAW1BR [THRL] / 4096)) * Vbandgap
3. Calculate the converted results to voltage:

   (STDR2 [IDATA] + (STDR2 [FDATA] / 4096)) * Vbandgap
4. Compare result from step 3 to THRH and THRL.S1 algorithm passes

   If (low threshold voltage <= converted voltage <= high threshold voltage).

36.5.10.2.6 Watchdog timer

The watchdog timer is an additional check that monitors the sequence of the self-testing algorithm implemented and checks that the algorithm completes within a safe time period. Watchdog timers can be enabled for CPU as well as CTU conversions. Each algorithm has a different watchdog timer that runs independently of the other. The watchdog timer for a particular algorithm can be enabled by setting STAWxR[WDTE]. The safe time value can be programmed in STBRR[WDT] (the default value is 10 ms, assuming an 80 MHz clock).

The safe time is measured starting from step 0 of the algorithm (including all normal chain conversions in between) to the point at which step 0 of the same algorithm starts again.

The sequence is as follows:

1. Program NCMR0 to select channels to be converted for normal conversion in Scan mode (MODE = 1).
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (14 h) (for 1 MSPS operation)

3. Select the self-testing algorithm in STCR3[ALG]. By default, all algorithms are selected; that is, all algorithms will be executed step-by-step, one after the other.
4. Enable the self-testing channel by setting STCR2[EN].
5. Program the safe period value in STBRR[WDT].
6. Enable the watchdog timer by setting STAWxR[WDTE]. Assume the setting of WDTE to be t0. (It is important to do all the programming first and to then enable WDTE as the safe time check is also performed between the setting of WDTE and the start of step 0 to check that algorithm has started within the safe time).
7. Start the normal conversion by setting MCR[NSTART].
8. After first chain conversion ends, step 0 of algorithm S is executed. Lets assume the start of STEP0 to be t1.
9. Step 1 and step 2 of algorithm S are executed.
10. The next chain conversions are performed.
11. When chain conversion completes, step 0 of C algorithm is performed.
12. After each chain conversion, a consecutive step of the C algorithm is performed. A similar sequence follows for algorithm C.
13. After the last step of the C algorithm is performed, another chain conversion is executed. At the end of this chain conversion, step 0 of algorithm S starts, repeating the whole sequence. Let's assume this time (starting of step 0) to be t2.
14. For algorithm S, if (t1 – t0) > Safe Period, or if (t2 – t1) > Safe Period, then the watchdog timer flags an error and STSR1[WDTERR] is set. A critical fault is asserted and an interrupt is also generated if enabled by STCR2[MSKWDTERR]. Otherwise, the watchdog timer counter is reset and starts again to monitor the same for the next sequence.
15. A similar sequence is followed for watchdog timers for algorithm C.

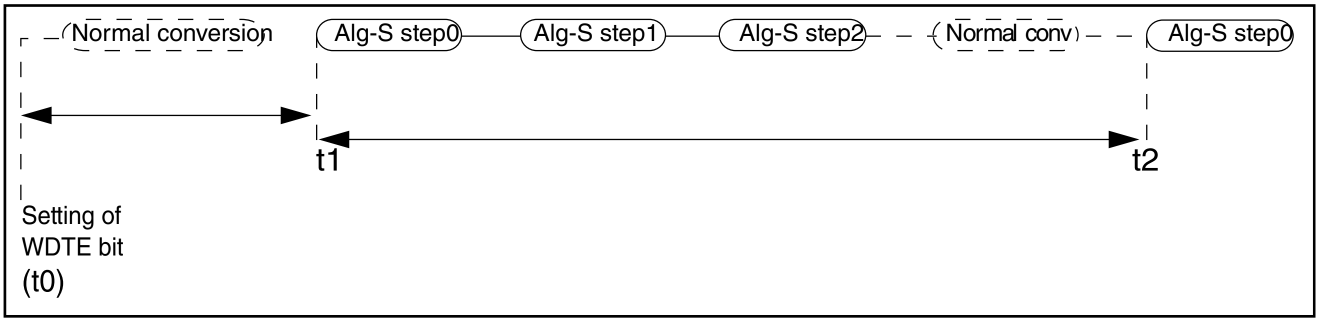


Figure 36-8. Watchdog timer monitor for algorithm S

Notes:

- As CTU does not incorporate any safe period checking mechanism, the watchdog timers can also be enabled for CTU conversions.
- Take care not to enable the watchdog timer for an algorithm that is not to be executed.

36.5.10.2.6.1 Watchdog sequence checking

The watchdog timer also incorporates sequence checking features that check the order of a particular algorithm's steps. If the steps are not in order, then an error is flagged by setting the STSR1[WDSERR]. A critical fault is asserted and an interrupt is also generated if enabled by STCR2[MSKWDSERR].

A watchdog sequence error is flagged in the following cases:

- If the steps of any algorithm are not executed in the proper order.
- If an abort chain occurs during a test channel conversion, that step has to be repeated at the end of the next chain. This will give a sequence error as soon as test channel conversion starts again.
    - Exception: If an abort chain occurs during the last step of algorithm S, then sequence errors are not flagged as the whole algorithm has to be repeated again.
- If, for CTU conversions, the step numbers provided by the CTU are not in order. Watchdog sequence checking is significant for CTU burst mode only.

If injected conversion occurs during the test channel, watchdog sequence errors are *not* flagged a lthough the ongoing step number is aborted and repeated again.

Note

The watchdog timer feature is applicable only for Scan mode, not for One-Shot mode.

36.5.10.2.7 Baud rate control for test channel

It defines the scheduling of the test channel between normal conversions. The scheduling rate is specified by STBRR[BR].

By default, if the test channel is enabled, one step of the selected algorithm is executed after every chain of normal conversion. The bandwidth consumed by the test channel depends on the number of channels in a normal chain. For example, if there are 100 normal conversions in a chain, then the test channel consumes only 1% of the total bandwidth, which is very small. However, if the number decreases to just four channels, then the bandwidth consumed by the test channel is 25%, which is significant (and may not be desirable as it slows down the normal conversion rate).

STBRR[BR] provides flexibility by scheduling the test channel conversion to be performed, not at the end of every chain, but at the end of BR + 1 number of chains. For example, if BR = 5, a single step of the selected algorithm for the test channel is performed after six chain conversions, then the next step is performed at end of the next six chain conversions, and so on. By default, the value of BR is 0.

Notes:

- This feature is applicable only for Scan mode, not for One-Shot mode. STBRR[BR] should be set to 0 for one shot mode.
- To use the baud rate feature in Scan mode, the NCMR register should have nonzero value.

36.5.10.2.7.1 Abort chain when baud rate is non-zero

As previously described, a nonzero value of STBRR[BR] means that the test channel conversion is performed at the end of BR + 1 number of chains. If an abort chain occurs during the chain in which the test channel is scheduled to be converted, then the test channel is converted after the next BR + 1 number of chains.

For example, if STBRR[BR] is programmed to 2, the sequence will be two normal chains (without any test channel conversion) followed by a chain with the test channel converted at the end. Now, if an abort chain occurs during first two chains it is treated as a normal chain abort and the test channel is converted at the end of the third chain only (as the case without any abort chain). But if the abort chain occurs during the third chain in which the test channel is scheduled to be converted, then the test channel is converted after next three chains (that is, at the end of sixth chain, counting from the beginning).

36.5.11 Conversion time

Total conversion time depends on ADC-SAR controller's operating clock. This clock can be prescaled to bus clock ÷ 2. The ADC-SAR operating clock (AD_clk) is always synchronous with bus clock. The frequency of AD_clk is dependent on settings of ADC_MCR[ADCLKSEL].

If ADC_MCR[ADCLKSEL] = 0, then AD_clk = (bus clock ÷ 2)

If ADC_MCR[ADCLKSEL] = 1, then AD_clk = bus clock

The various components of conversion time and affecting configurations are shown and described in Figure 36-9.

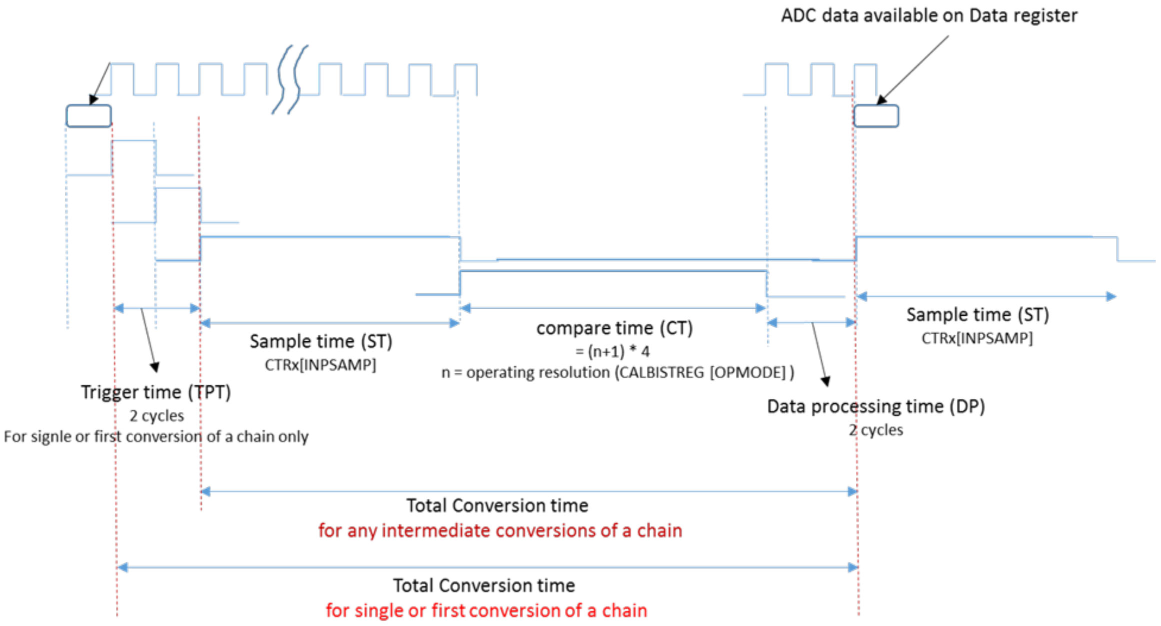

ADC data available on Data register
Sample time (ST)
CTRx[INPSAMP]
compare time (CT)
= (n+1) * 4
n = operating resolution (CALBISTREG [OPMODE])
Sample time (ST)
CTRx[INPSAMP]
Trigger time (TPT)
2 cycles
For signle or first conversion of a chain only
Data processing time (DP)
2 cycles
Total Conversion time
for any intermediate conversions of a chain
Total Conversion time
for single or first conversion of a chain

Figure 36-9. Conversion time

- Trigger processing time (TPT):

  To prepare the channel and start the operation, two clock cycles of bus clock is required by the ADC for the first conversion and for subsequent conversions in a chain, or for a continuous conversion, this time is not required as it is hidden in the pipeline operation. Triggers from Synchronous CTU interface requires two cycle of bus clock for first conversion and for back-to-back CTU conversions it takes only one cycle for trigger processing as the other cycle is hidden in pipeline operation. Triggers from Asynchronous CTU interface requires three cycles of bus clock for synchronization and processing.
- Sample phase time (ST):

  Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing Registers ADC_CTR*x* (*x* = 0...2) for different types of channels. Value in each register affects the sample time in similar way. The value in the register represents units of cycle of the AD_clk. The minium value of sample time is 8. If the value programmed is less than 8, then it has no effect on sample time duration and in this case sample time will be 8 AD_clk cycles.
- Compare phase time (CT):

  The compare phase of the ADC is dependent on the resolution setting. It takes ((n + 1) × 4) cycles of AD_clk, where n is the resolution setting configured in CALBISTREG[OPMODE].

- Data processing time (DP):

The ADC takes 2 cycles of AD_clk to post process and load the data int registers.

- Total Conversion Time:

  The total time required for single or first conversion is [TPT+ST+CT+DP] cycles of AD_clk and for subsequent conversion is a chain or continuous conversion is [ST+CT+DP] cycles of AD_clk.

  Examples:

  - CALBISTREG[OPMODE] = 110b; CTR*x*[INPSAMP] = 20d; software/hardware triggered (NSTART):
    - Single/First Conversion: 2 + 20 + ((13 + 1) × 4) + 2 = 2 + 20 + 56 + 2 = 80 cycles of AD_clk
    - Subsequent/continuous conversion: 20 + ((13 + 1) × 4) + 2 = 20 + 56 + 2 = 78 cycles of AD_clk
  - CALBISTREG[OPMODE] = 001b; CTR*x*[INPSAMP] = 22d; software/hardware triggered (NSTART):
    - Single/First Conversion: 2 + 22 + ((12 + 1) × 4) + 2 = 2 + 22 + 52 + 2 = 78 cycles of AD_clk
    - Subsequent/continuous conversion: 22 + ((12 + 1) × 4) + 2 = 22 + 52 + 2 = 76 cycles of AD_clk
  - CALBISTREG[OPMODE] = 110b; CTR*x*[INPSAMP] = 20d; CTU triggered:
    - Single/First Conversion: 2 + 20 + ((13 + 1) × 4) + 2 = 2 + 20 + 56 + 2 = 80 cycles of AD_clk
    - Subsequent/continuous conversion: 1 + 20 + ((13 + 1) × 4) + 2 = 1 + 20 + 56 + 2 = 79 cycles of AD_clk
  - CALBISTREG[OPMODE] = 001b; CTR*x*[INPSAMP] = 22d; CTU triggered:
    - Single/First Conversion: 2 + 22 + ((12 + 1) × 4) + 2 = 2 + 22 + 52 + 2 = 78 cycles of AD_clk
    - Subsequent/continuous conversion: 1 + 22 + ((12 + 1) × 4) + 2 = 1 + 22 + 52 + 2 = 77 cycles of AD_clk
- Pre-Sampling:

  When pre-sampling is enabled, the ADC does pre-sampling prior to starting the actual channel sampling as described in pre-sampling section. This phase takes CTR*x*[INPSAMP] cycles of AD_clk. Additionally, to switch from pre-sample phase to actual channel sampling phase, it takes another two cycles of AD_clk. So, total CTR*x*[INPSAMP] + 2 cycles additionally required for a conversion to be completed. Thus, in case of pre-sample, all the equations and examples above are valid with the addition of (CTR*x*[INPSAMP] + 2) cycles.

36.5.12 Conversion data processing

The raw converted ADC data contains many types of errors such as offset, gain, DC bias, and so on. Thus, to generate error-free results, raw converted data is processed before it is written to a result register. The process of error correction goes bit-by-bit during conversion with the values generated during the offset calculation and calibration process. This helps in reducing final data processing time as only one addition/addition (2's complement) is performed parallely with conversion.

36.6 Clock frequency

The ADC internal blocks are designed to work with an 80 MHz clock. Thus, the specified speed can be met with this clock frequency. If the clock is faster than this, ADCA cannot work properly and the ADC may malfunction. If the clock is slower, then the ADC speed will be lower, but it will work properly. It should be noted that there is an absolute maximum value of the sampling time that the analog portion of the ADC can support. For slower clocks, the sampling clock count will need to be programmed accordingly.

Chapter 37
Temperature Sensor (TSENS)

37.1 Introduction

The device includes two on-chip temperature sensors. The temperature sensor module monitors device temperature and delivers one analog output signal and three digital output signals each (under- and over-temperature flags). The analog output consists of a voltage signal directly proportional to the internal junction temperature. The analog output is connected to an input channel of an ADC on the device (see the ADC details for the channel number). The internal junction temperature must be calculated by software based on the sampled temperature sensor voltage; sampled bandgap reference voltage, which comes from another module; and calibration parameter values stored in internal flash memory. The three digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions.

The digital outputs signal when the junction temperature drifts below the low temperature threshold or above one of the two high temperature thresholds. These signals notify the device to take action to appropriately adjust the device temperature in response to an out-of-specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

37.2 Functional Description

The temperature sensor generates one analog output voltage, V_{TSENS}(T), which is proportional to the absolute current junction temperature of the device, and three digital outputs that signal whether the junction temperature has reached either a pre-set low temperature threshold or one of two pre-set high temperature thresholds.

An on-chip ADC module is used to convert the analog output voltage, as well as the bandgap reference voltage, into a digital representation. These values, along with parameter values stored in on-chip flash memory, are used by software to calculate the device junction temperature.

37.2.1 Temperature threshold detection (digital output generation)

Temperature threshold trimming values are adjusted through calibration during factory test and stored in on-chip flash memory. While enabled, if a new trimming value is loaded into the Temperature Sensor module, it may take up to 1 microsecond for the digital outputs to be updated to reflect the new trimming condition.

When the temperature threshold detection feature is enabled, the temperature sensor monitors the internal junction temperature of the chip and asserts a signal if any of the following temperature thresholds are crossed.

- The low temperature digital output signals if the junction temperature falls below the low temperature threshold (-40°C).
- The high temperature digital output 1 signals if the junction temperature rises above the first high temperature threshold (150°C).
- The high temperature digital output 2 signals if the junction temperature rises above the second high temperature threshold (165°C).

NOTE

Some devices do not support 165°C operation.

The threshold detection circuit is calibrated with respect to the factory-test temperatures: the low temperature threshold is trimmed to match the cold insertion test temperature (nominally equal to –40°C), and the high temperature thresholds are trimmed such that one of them matches the hot insertion test temperature (nominally equal to 150°C or 165°C). In other words, the thresholds are not trimmed to absolute values of temperature but to the specific temperatures for which the device is validated during factory test.

Hysteresis is applied whenever outputs transition to a logic level low in order to eliminate spurious toggling. Digital output behavior is illustrated on Figure 37-1.

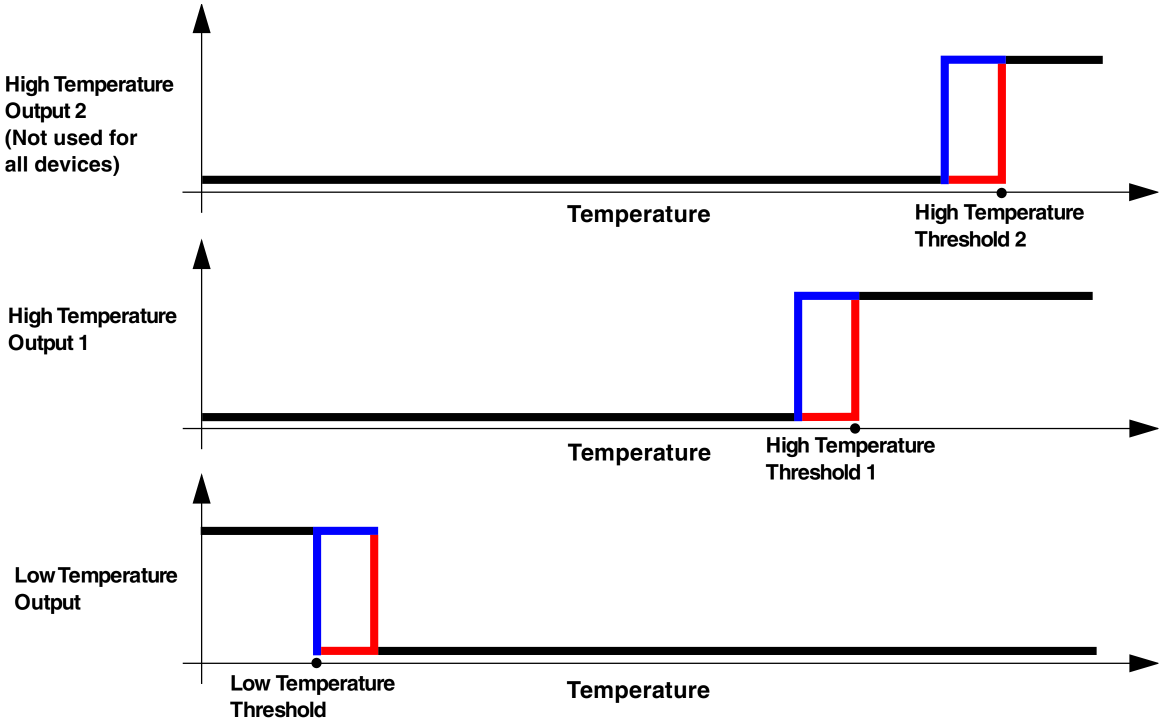

High Temperature Output 2 (Not used for all devices)
Temperature
High Temperature Threshold 2
High Temperature Output 1
Temperature
High Temperature Threshold 1
Low Temperature Output
Low Temperature Threshold
Temperature

Figure 37-1. Digital output behavior with temperature

37.2.2 Linear temperature sensor (analog output generation)

When analog output generation is enabled, the temperature sensor outputs a voltage proportional to the internal junction temperature of the chip. This analog voltage signal is converted into a digital code by an on-chip ADC. The temperature value is obtained from a linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programmed into flash memory.

37.3 Temperature formula

The system chain that translates device junction temperature into a digital variable is composed of the temperature sensor, a bandgap reference voltage source and the on-chip ADC. Both analog output voltages of the temperature sensor and the bandgap reference voltage source must be converted by the ADC into digital codes to obtain the device junction temperature. The temperature formula shown in Figure 37-2 is used to calculate internal device junction temperature.

$$T \times 2^{2} = \frac{(K_{1} \times V_{BG\_CODE}(T)) \times 2^{-1} + (K_{2} \times V_{TSENS\_CODE}(T)) \times 2^{3}}{[(K_{3} \times V_{BG\_CODE}(T)) \times 2^{2} + (K_{4} \times V_{TSENS\_CODE}(T))] \times 2^{-10}} \quad [K]$$

$$V_{TSENS\_CODE}(T) = \frac{V_{TSENS}(T)}{V_{ref}} \times 2^{12} \qquad V_{BG\_CODE}(T) = \frac{V_{BG}(T)}{V_{ref}} \times 2^{12}$$

- T is the internal junction temperature in Kelvin
- $V_{TSENS}(T)$ is the temperature sensor output sampled by the ADC
- $V_{BG}(T)$ is the bandgap reference voltage sampled by the ADC
- V_{ref} is the ADC reference voltage in normal usage of the device
- K1, K2, K3, and K4 are 16-bit signed integer constants in two's-complement representation stored in flash memory (obtained from factory calibration)

Figure 37-2. Temperature formula

Parameters K1, K2, K3, and K4 are 16-bit signed integer constants in two's-complement representation that are stored in flash memory during factory test and calibration. These constants must be used in the temperature formula to calculate device junction temperature.

37.4 Calculating device temperature

Software must perform the following steps in order to calculate device temperature.

1. Measure the bandgap reference voltage using the on-chip ADC module and calculate $V_{BG\_CODE}(T)$ according to the formula shown in Figure 37-2.
2. Measure the temperature sensor output voltage using the on-chip ADC module and calculate $V_{TSENS\_CODE}(T)$ according to the formula shown in Figure 37-2
3. Retrieve parameters K1, K2, K3, and K4 from flash memory.
4. Calculate T according to the formula shown in Figure 37-2.

Note

Temperature Formula may be calculated using signed 32-bit (integer) operations showing negligible precision loss.

Chapter 38
Sine Wave Generator (SGEN)

38.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Sine Wave Generator (SGEN) generates a high-quality sinusoidal voltage signal. It can be programmed with the desired oscillation frequency and amplitude voltage. A wide frequency range (1 kHz–50 kHz in 16 Hz steps) is easily programmable through a simple register interface. The linearity and noise performance is carefully optimized through digital processing.

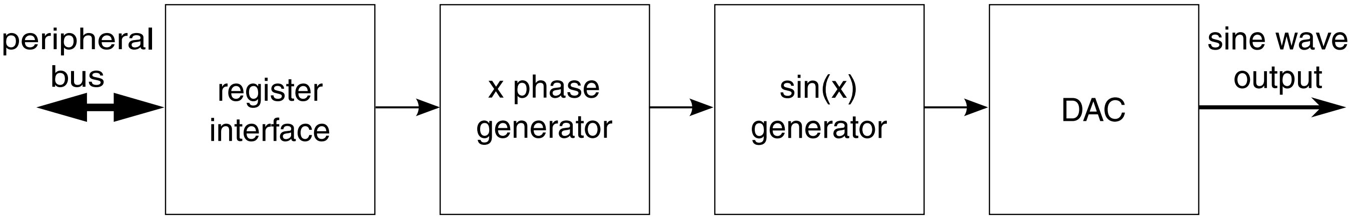


Figure 38-1. SGEN block diagram

38.2 Features

- Input clock frequency range: 12 MHz–20 MHz
- Output sinusoidal signal:
  - Frequency range: 1 kHz–50 kHz
  - Peak-to-peak amplitude: user adjustable

38.3 Memory map and register description

The SGEN memory map is shown below followed by the description for each register and its fields.

SGEN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (SGEN_CTRL)	32	R/W	0001_0000h	38.3.1/1068
4	Status Register (SGEN_STAT)	32	R/W	0000_0000h	38.3.2/1070

38.3.1 Control Register (SGEN_CTRL)

The SGEN_CTRL register enables the following operations:

- Control and read the output sine wave frequency.
- Control and read the output sine wave amplitude.
- Enable and disable the input trigger to align the phase of the sine wave output.
- Select the active trigger input for phase alignment.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LDOS	0	IOAMPL				0		SEMASK	0			TRIG_SEL	TRIG_EN	Reserved	PDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IOFREQ															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SGEN_CTRL field descriptions

Field	Description
0 LDOS	Load sine wave frequency SGEN_CTRL[IOFREQ] is loaded by waiting until LDOS = 0 and then setting LDOS to 1. 0 Wait for I/O sine wave frequency 1 Load I/O sine wave frequency

Table continues on the next page...

SGEN_CTRL field descriptions (continued)

Field	Description
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–5 IOAMPL	IO sine wave amplitude Controls the sine wave voltage amplitude on the output pin. See Output sine wave amplitude for further information.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SEMASK	Sine wave generator error mask interrupt register 0 Mask the SGEN error interrupt source 1 Enable the SGEN error interrupt source
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 TRIG_SEL	Input trigger select This field selects the active trigger input when SGEN_CTRL[TRIG_EN] is enabled. 0 First trigger input will used for aligning the sine wave output. 1 Second trigger input will used for aligning the sine wave output.
13 TRIG_EN	Input phase align trigger enable bit 0 Trigger inputs will not affect the sine wave output. 1 Trigger input based on TRIG_SEL field will be used to align the sine wave output. The sine wave appears at the output after the first active trigger occurs. Its frequency is determined by IOFreq.
14 Reserved	This field is reserved.
15 PDS	Enter/exit Power Down mode 0 Force SGEN to exit Power Down mode. 1 Force SGEN to enter Power Down mode.
16–31 IOFREQ	Output sine wave frequency See Output sine wave amplitude to determine value for this field.

38.3.2 Status Register (SGEN_STAT)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								SERR	0			FERR	PHERR	0	
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SGEN_STAT field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SERR	Error interrupt status This field is set only by the chip and not by software. It can be cleared in software by writing a 1 to it. During an SGEN_STAT[SERR] clear operation, the hardware set is not allowed to set this field. **NOTE:** This field is set on either of the following conditions: • There is an error in the demodulator block. • The input trigger's phase is out of the 1% window of the current sine wave output. 0 No error interrupt pending 1 Error interrupt pending
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 FERR	Force error interrupt **NOTE:** Because this field is set in the bus clock domain and the SGEN digital block operates in the SGEN clock domain, both clocks must be present to set the SGEN_STAT[SERR] field. Also, due to the synchronization between two asynchronous clocks, the setting of SGEN_STAT[SERR] will

Table continues on the next page...

SGEN_STAT field descriptions (continued)

Field	Description
	occur, at maximum, 4 bus clock and 4 SGEN clock cycles after the setting of SGEN_STAT[FERR]. 0 An error interrupt will not be forced. 1 An error interrupt will be forced.
13 PHERR	Phase error SERR will be set when this field transistions from 0 to 1. This field can be reset to 0 by software.
14–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

38.4 Functional description

38.4.1 Operation after a power-on reset

After a power-on reset, the SGEN produces a DC value (no sinusoidal fluctuations).

38.4.2 Output sine wave frequency

The output sine wave frequency *f* is controlled by SGEN_CTRL[IOFREQ] according to Equation 16 on page 1071 where frequencies *f* and InputFrequency are in Hz, and IOFREQ is in decimal format. IOFREQ must be chosen to ensure that *f* remains between 1 kHz and 50 kHz.

$$f = \frac{\mathrm{InputFrequency}}{1048576} \times \mathrm{IOFREQ}$$

Equation 16. Output sine wave frequency

38.4.3 Output sine wave amplitude

The amplitude of the output sine wave is controlled by SGEN_CTRL[IOAMPL] as described in the following table.

Table 38-1. Resulting sine wave amplitude for each IOAMPL value

IOAMPL value (decimal)	Sine wave amplitude (V)	Sine wave amplitude step (V)
0	0.423	
1	0.487	0.064

Table continues on the next page...

Table 38-1. Resulting sine wave amplitude for each IOAMPL value (continued)

IOAMPL value (decimal)	Sine wave amplitude (V)	Sine wave amplitude step (V)
2	0.550	0.063
3	0.613	0.063
4	0.675	0.062
5	0.737	0.062
6	0.797	0.060
7	0.857	0.060
8	0.987	0.130
9	1.136	0.149
10	1.284	0.148
11	1.430	0.146
12	1.575	0.145
13	1.719	0.144
14	1.861	0.142
15	2.00	0.139

38.5 Initialization and application information

38.5.1 Changing the output frequency

To change the output frequency:

1. Determine the required value of SGEN_CTRL[IOFREQ] using Equation 16 on page 1071.
2. Ensure that SGEN_CTRL[LDOS] = 0.
3. Write the new value of the SGEN_CTRL[IOFREQ] field.
4. Set SGEN_CTRL[LDOS].

The SGEN will begin producing the new sine wave frequency two input clock cycles after this procedure is completed.

38.5.2 Phase control of the sine wave output

The normal output of the sine wave generator is a free-running sine wave with the amplitude and frequency determined by the SGEN_CTRL register settings. SGEN_CTRL[TRIG_EN] enables the user to phase align the sine wave output to one of the trigger inputs, which is selected by SGEN_CTRL[TRIG_SEL]. The trigger input is a periodic signal which has the same fequency as the sine wave output but might have a different phase, either leading or lagging. The phase of the trigger input signal is compared to the phase the sine wave output. If a phase offset is detected, the sine wave output is realigned so that the phase difference becomes zero, beginning with the next sine wave cycle.

If the phase difference of the trigger input varies more than 1%, either leading or lagging, from sine wave output, no phase adjustment will take place. Instead, SGEN_STAT[SERR] will be set indicating that the phase offset has exceeded 1%. In this case, the sine wave output will not be re-generated.

NOTE

- Because the trigger input is an asynchronous signal, it must be held high for at least 2 SGEN clock cycles in order to capture the input trigger. This requirement must be guaranteed by the module generating the trigger input signal. If this requirement is not met, phase alignment will not function as expected.
- When SGEN_CTRL[TRIG_EN] is set, the sine wave output will not be generated until the first trigger occurs.
- If the actual trigger position falls in the +/- 1% range, the sine wave output will be aligned to the trigger beginning with the next period.

38.5.3 Preserving the SGEN_CTRL data

Changes made to the SGEN_CTRL register must be preserved for at least 50 clock cycles. Failure to do so might cause unexpected output from the SGEN.

Chapter 39
Enhanced Motor Control Timer (eTimer)

39.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

This section briefly describes the eTimer, its features and customization, and contains block diagrams.

39.1.1 Overview

An eTimer module provides:

- Six identical counter/timer channels
- One watchdog timer function (might not be available on every eTimer module instance)

Each 16-bit counter/timer channel contains a prescaler, a counter, a load register, a hold register, two queued capture registers, two compare registers, two compare preload registers, and four control registers.

Note

> This document uses the terms "Timer" and "Counter" interchangeably because the counter/timers may perform either or both tasks.

The Load register provides the initialization value to the counter when the counter's terminal value has been reached. For true modulo counting the counter can also be initialized by the CMPLD1 or CMPLD2 registers.

The Hold register captures the counter's value when other counters are being read. This feature supports the reading of cascaded counters coherently.

The Capture registers enable an external signal to take a "snap shot" of the counter's current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The Prescaler provides different time bases useful for clocking the counter/timer.

The Counter provides the ability to count internal or external events.

Within the eTimer module, the input pins are shareable.

39.1.2 Features

The eTimer module design includes these distinctive features:

- 16-bit counters/timers: Six.
- Count up/down.
- Counters are cascadable.
- Enhanced programmable up/down modulo counting.
- Max count rate equals peripheral clock/2 for external clocks.
- Max count rate equals peripheral clock for internal clocks.
- Count once or repeatedly.
- Counters are preloadable.
- Compare registers are preloadable.
- Counters can share available input pins.
- Separate prescaler for each counter.
- Each counter has capture and compare capability.
- Continuous and single shot capture for enhanced speed measurement.
- DMA support of capture registers and compare registers.
- 32-bit watchdog capability to detect stalled quadrature counting (might not be available on every eTimer module instance).
- OFLAG comparison for safety critical applications.
- Programmable operation during debug mode and stop mode.
- Programmable input filter.
- Counting start can be synchronized across counters.

39.1.3 Module block diagram

The eTimer block diagram is shown in the following figure.

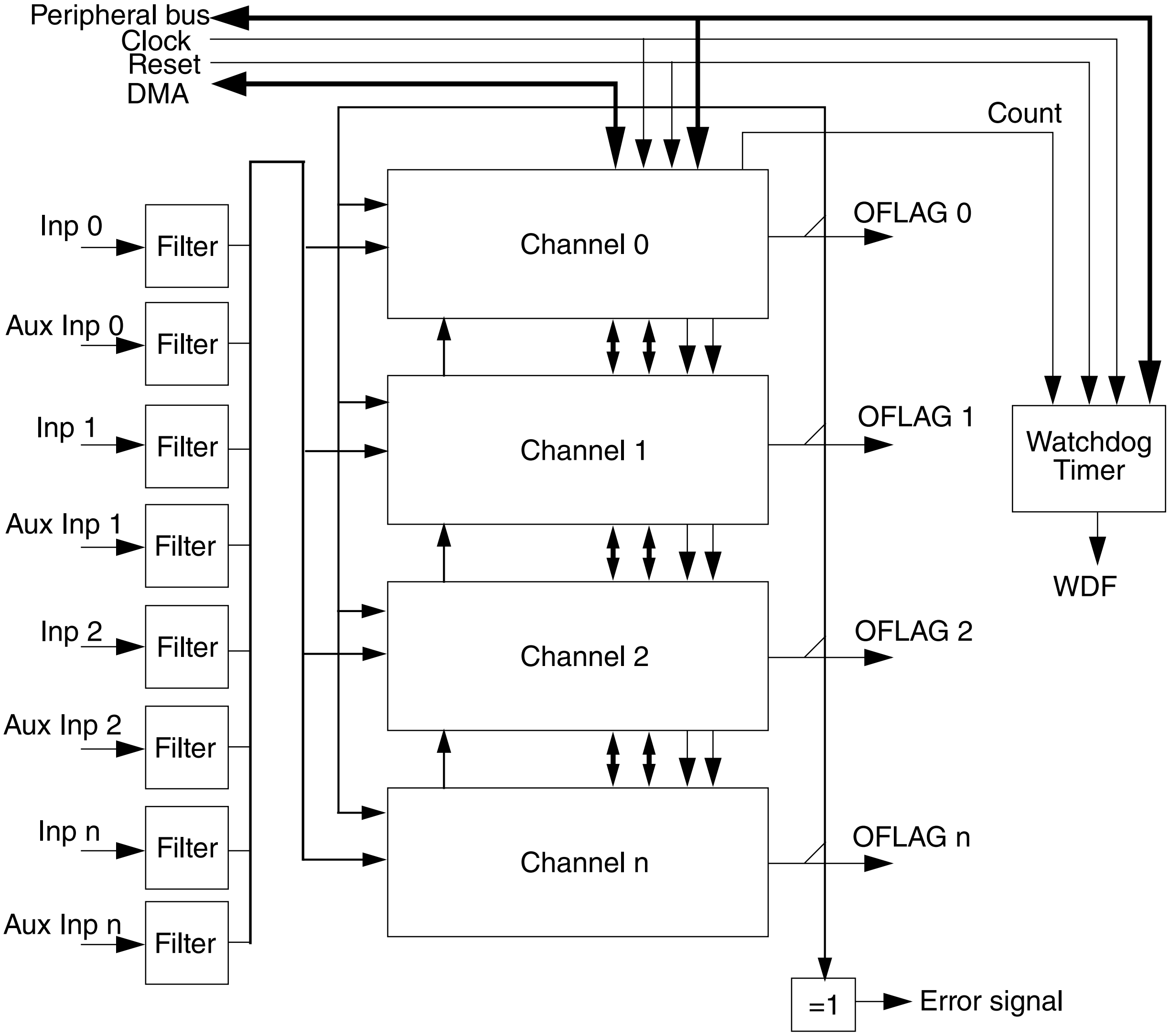


Figure 39-1. eTimer Block Diagram

39.1.4 Channel Block Diagram

Each of the timer/counter channels within the eTimer are shown in the following figure.

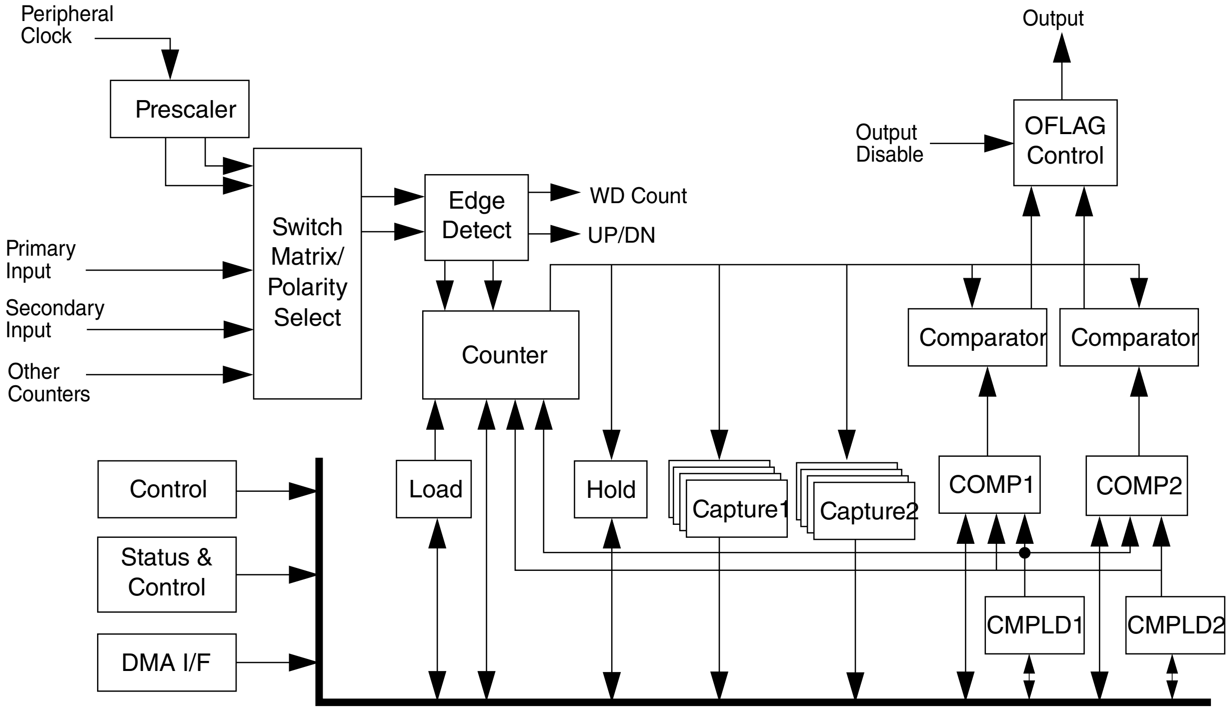


Figure 39-2. eTimer Channel Block Diagram

39.2 External Signal Descriptions

Each eTimer module has six external signals that can be used as either inputs or outputs. The number of auxilliary inputs per module is device-dependent. See the configuration section for more information. The eTimer also interfaces to the Peripheral Bus.

39.2.1 TIO[n:0] - Timer Input/Outputs

These pins can be independently configured to be either timer input sources or output flags. In Figure 39-2, the TIO signals are shown as the Primary Inputs, and the Outputs from the OFLAG Control block.

39.2.2 TAI[n:0] - Timer Auxiliary Inputs

These pins act as alternate input choices for the timer channels. The TAI signals are shown as Aux Inp 1..n in .

39.3 Memory map and register definition

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level and the address offset is defined at the module level. There are a set of registers for each timer channel, a set for the watchdog timer, and a set of configuration registers. Certain registers are labelled as not byte accessible. However, no error is generated if a byte access occurs.

The base address of the Watchdog Timer registers is equal to the base address of the eTimer plus an offset of 100h.

ETIMER memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Channel n Compare Register 1 (ETIMER_CH0_COMP1)	16	R/W	0000h	39.3.1/1083
2	Channel n Compare Register 2 (ETIMER_CH0_COMP2)	16	R/W	0000h	39.3.2/1083
4	Channel n Capture Register 1 (ETIMER_CH0_CAPT1)	16	R	0000h	39.3.3/1084
6	Channel n Capture Register 2 (ETIMER_CH0_CAPT2)	16	R	0000h	39.3.4/1084
8	Channel n Load Register (ETIMER_CH0_LOAD)	16	R/W	0000h	39.3.5/1085
A	Channel n Hold Register (ETIMER_CH0_HOLD)	16	R	0000h	39.3.6/1085
C	Channel n Counter Register (ETIMER_CH0_CNTR)	16	R/W	0000h	39.3.7/1086
E	Channel n Control Register 1 (ETIMER_CH0_CTRL1)	16	R/W	0000h	39.3.8/1086
10	Channel n Control Register 2 (ETIMER_CH0_CTRL2)	16	R/W	0000h	39.3.9/1089
12	Channel n Control Register 3 (ETIMER_CH0_CTRL3)	16	R/W	0F00h	39.3.10/ 1091
14	Channel n Status Register (ETIMER_CH0_STS)	16	R/W	0000h	39.3.11/ 1092
16	Channel n Interrupt and DMA Enable Register (ETIMER_CH0_INTDMA)	16	R/W	0000h	39.3.12/ 1094
18	Channel n Comparator Load Register 1 (ETIMER_CH0_CMPLD1)	16	R/W	0000h	39.3.13/ 1095
1A	Channel n Comparator Load Register 2 (ETIMER_CH0_CMPLD2)	16	R/W	0000h	39.3.14/ 1096
1C	Channel n Compare and Capture Control Register (ETIMER_CH0_CCCTRL)	16	R/W	0000h	39.3.15/ 1096
1E	Channel n Input Filter Register (ETIMER_CH0_FILT)	16	R/W	0000h	39.3.16/ 1098
20	Channel n Compare Register 1 (ETIMER_CH1_COMP1)	16	R/W	0000h	39.3.1/1083
22	Channel n Compare Register 2 (ETIMER_CH1_COMP2)	16	R/W	0000h	39.3.2/1083
24	Channel n Capture Register 1 (ETIMER_CH1_CAPT1)	16	R	0000h	39.3.3/1084
26	Channel n Capture Register 2 (ETIMER_CH1_CAPT2)	16	R	0000h	39.3.4/1084

Table continues on the next page...

ETIMER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	Channel n Load Register (ETIMER_CH1_LOAD)	16	R/W	0000h	39.3.5/1085
2A	Channel n Hold Register (ETIMER_CH1_HOLD)	16	R	0000h	39.3.6/1085
2C	Channel n Counter Register (ETIMER_CH1_CNTR)	16	R/W	0000h	39.3.7/1086
2E	Channel n Control Register 1 (ETIMER_CH1_CTRL1)	16	R/W	0000h	39.3.8/1086
30	Channel n Control Register 2 (ETIMER_CH1_CTRL2)	16	R/W	0000h	39.3.9/1089
32	Channel n Control Register 3 (ETIMER_CH1_CTRL3)	16	R/W	0F00h	39.3.10/ 1091
34	Channel n Status Register (ETIMER_CH1_STS)	16	R/W	0000h	39.3.11/ 1092
36	Channel n Interrupt and DMA Enable Register (ETIMER_CH1_INTDMA)	16	R/W	0000h	39.3.12/ 1094
38	Channel n Comparator Load Register 1 (ETIMER_CH1_CMPLD1)	16	R/W	0000h	39.3.13/ 1095
3A	Channel n Comparator Load Register 2 (ETIMER_CH1_CMPLD2)	16	R/W	0000h	39.3.14/ 1096
3C	Channel n Compare and Capture Control Register (ETIMER_CH1_CCCTRL)	16	R/W	0000h	39.3.15/ 1096
3E	Channel n Input Filter Register (ETIMER_CH1_FILT)	16	R/W	0000h	39.3.16/ 1098
40	Channel n Compare Register 1 (ETIMER_CH2_COMP1)	16	R/W	0000h	39.3.1/1083
42	Channel n Compare Register 2 (ETIMER_CH2_COMP2)	16	R/W	0000h	39.3.2/1083
44	Channel n Capture Register 1 (ETIMER_CH2_CAPT1)	16	R	0000h	39.3.3/1084
46	Channel n Capture Register 2 (ETIMER_CH2_CAPT2)	16	R	0000h	39.3.4/1084
48	Channel n Load Register (ETIMER_CH2_LOAD)	16	R/W	0000h	39.3.5/1085
4A	Channel n Hold Register (ETIMER_CH2_HOLD)	16	R	0000h	39.3.6/1085
4C	Channel n Counter Register (ETIMER_CH2_CNTR)	16	R/W	0000h	39.3.7/1086
4E	Channel n Control Register 1 (ETIMER_CH2_CTRL1)	16	R/W	0000h	39.3.8/1086
50	Channel n Control Register 2 (ETIMER_CH2_CTRL2)	16	R/W	0000h	39.3.9/1089
52	Channel n Control Register 3 (ETIMER_CH2_CTRL3)	16	R/W	0F00h	39.3.10/ 1091
54	Channel n Status Register (ETIMER_CH2_STS)	16	R/W	0000h	39.3.11/ 1092
56	Channel n Interrupt and DMA Enable Register (ETIMER_CH2_INTDMA)	16	R/W	0000h	39.3.12/ 1094
58	Channel n Comparator Load Register 1 (ETIMER_CH2_CMPLD1)	16	R/W	0000h	39.3.13/ 1095
5A	Channel n Comparator Load Register 2 (ETIMER_CH2_CMPLD2)	16	R/W	0000h	39.3.14/ 1096
5C	Channel n Compare and Capture Control Register (ETIMER_CH2_CCCTRL)	16	R/W	0000h	39.3.15/ 1096
5E	Channel n Input Filter Register (ETIMER_CH2_FILT)	16	R/W	0000h	39.3.16/ 1098
60	Channel n Compare Register 1 (ETIMER_CH3_COMP1)	16	R/W	0000h	39.3.1/1083

Table continues on the next page...

ETIMER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
62	Channel n Compare Register 2 (ETIMER_CH3_COMP2)	16	R/W	0000h	39.3.2/1083
64	Channel n Capture Register 1 (ETIMER_CH3_CAPT1)	16	R	0000h	39.3.3/1084
66	Channel n Capture Register 2 (ETIMER_CH3_CAPT2)	16	R	0000h	39.3.4/1084
68	Channel n Load Register (ETIMER_CH3_LOAD)	16	R/W	0000h	39.3.5/1085
6A	Channel n Hold Register (ETIMER_CH3_HOLD)	16	R	0000h	39.3.6/1085
6C	Channel n Counter Register (ETIMER_CH3_CNTR)	16	R/W	0000h	39.3.7/1086
6E	Channel n Control Register 1 (ETIMER_CH3_CTRL1)	16	R/W	0000h	39.3.8/1086
70	Channel n Control Register 2 (ETIMER_CH3_CTRL2)	16	R/W	0000h	39.3.9/1089
72	Channel n Control Register 3 (ETIMER_CH3_CTRL3)	16	R/W	0F00h	39.3.10/ 1091
74	Channel n Status Register (ETIMER_CH3_STS)	16	R/W	0000h	39.3.11/ 1092
76	Channel n Interrupt and DMA Enable Register (ETIMER_CH3_INTDMA)	16	R/W	0000h	39.3.12/ 1094
78	Channel n Comparator Load Register 1 (ETIMER_CH3_CMPLD1)	16	R/W	0000h	39.3.13/ 1095
7A	Channel n Comparator Load Register 2 (ETIMER_CH3_CMPLD2)	16	R/W	0000h	39.3.14/ 1096
7C	Channel n Compare and Capture Control Register (ETIMER_CH3_CCCTRL)	16	R/W	0000h	39.3.15/ 1096
7E	Channel n Input Filter Register (ETIMER_CH3_FILT)	16	R/W	0000h	39.3.16/ 1098
80	Channel n Compare Register 1 (ETIMER_CH4_COMP1)	16	R/W	0000h	39.3.1/1083
82	Channel n Compare Register 2 (ETIMER_CH4_COMP2)	16	R/W	0000h	39.3.2/1083
84	Channel n Capture Register 1 (ETIMER_CH4_CAPT1)	16	R	0000h	39.3.3/1084
86	Channel n Capture Register 2 (ETIMER_CH4_CAPT2)	16	R	0000h	39.3.4/1084
88	Channel n Load Register (ETIMER_CH4_LOAD)	16	R/W	0000h	39.3.5/1085
8A	Channel n Hold Register (ETIMER_CH4_HOLD)	16	R	0000h	39.3.6/1085
8C	Channel n Counter Register (ETIMER_CH4_CNTR)	16	R/W	0000h	39.3.7/1086
8E	Channel n Control Register 1 (ETIMER_CH4_CTRL1)	16	R/W	0000h	39.3.8/1086
90	Channel n Control Register 2 (ETIMER_CH4_CTRL2)	16	R/W	0000h	39.3.9/1089
92	Channel n Control Register 3 (ETIMER_CH4_CTRL3)	16	R/W	0F00h	39.3.10/ 1091
94	Channel n Status Register (ETIMER_CH4_STS)	16	R/W	0000h	39.3.11/ 1092
96	Channel n Interrupt and DMA Enable Register (ETIMER_CH4_INTDMA)	16	R/W	0000h	39.3.12/ 1094
98	Channel n Comparator Load Register 1 (ETIMER_CH4_CMPLD1)	16	R/W	0000h	39.3.13/ 1095
9A	Channel n Comparator Load Register 2 (ETIMER_CH4_CMPLD2)	16	R/W	0000h	39.3.14/ 1096

Table continues on the next page...

ETIMER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C	Channel n Compare and Capture Control Register (ETIMER_CH4_CCCTRL)	16	R/W	0000h	39.3.15/ 1096
9E	Channel n Input Filter Register (ETIMER_CH4_FILT)	16	R/W	0000h	39.3.16/ 1098
A0	Channel n Compare Register 1 (ETIMER_CH5_COMP1)	16	R/W	0000h	39.3.1/1083
A2	Channel n Compare Register 2 (ETIMER_CH5_COMP2)	16	R/W	0000h	39.3.2/1083
A4	Channel n Capture Register 1 (ETIMER_CH5_CAPT1)	16	R	0000h	39.3.3/1084
A6	Channel n Capture Register 2 (ETIMER_CH5_CAPT2)	16	R	0000h	39.3.4/1084
A8	Channel n Load Register (ETIMER_CH5_LOAD)	16	R/W	0000h	39.3.5/1085
AA	Channel n Hold Register (ETIMER_CH5_HOLD)	16	R	0000h	39.3.6/1085
AC	Channel n Counter Register (ETIMER_CH5_CNTR)	16	R/W	0000h	39.3.7/1086
AE	Channel n Control Register 1 (ETIMER_CH5_CTRL1)	16	R/W	0000h	39.3.8/1086
B0	Channel n Control Register 2 (ETIMER_CH5_CTRL2)	16	R/W	0000h	39.3.9/1089
B2	Channel n Control Register 3 (ETIMER_CH5_CTRL3)	16	R/W	0F00h	39.3.10/ 1091
B4	Channel n Status Register (ETIMER_CH5_STS)	16	R/W	0000h	39.3.11/ 1092
B6	Channel n Interrupt and DMA Enable Register (ETIMER_CH5_INTDMA)	16	R/W	0000h	39.3.12/ 1094
B8	Channel n Comparator Load Register 1 (ETIMER_CH5_CMPLD1)	16	R/W	0000h	39.3.13/ 1095
BA	Channel n Comparator Load Register 2 (ETIMER_CH5_CMPLD2)	16	R/W	0000h	39.3.14/ 1096
BC	Channel n Compare and Capture Control Register (ETIMER_CH5_CCCTRL)	16	R/W	0000h	39.3.15/ 1096
BE	Channel n Input Filter Register (ETIMER_CH5_FILT)	16	R/W	0000h	39.3.16/ 1098
100	Watchdog Time-out Low Word Register (ETIMER_WDTOL)	16	R/W	0000h	39.3.17/ 1099
102	Watchdog Time-out High Word Register (ETIMER_WDTOH)	16	R/W	0000h	39.3.18/ 1100
10C	Channel Enable Register (ETIMER_ENBL)	16	R/W	003Fh	39.3.19/ 1100
110	DMA Request 0 Select Register (ETIMER_DREQ0)	16	R/W	0000h	39.3.20/ 1101
112	DMA Request 1 Select Register (ETIMER_DREQ1)	16	R/W	0000h	39.3.21/ 1102
114	DMA Request 2 Select Register (ETIMER_DREQ2)	16	R/W	0000h	39.3.22/ 1104
116	DMA Request 3 Select Register (ETIMER_DREQ3)	16	R/W	0000h	39.3.23/ 1106

39.3.1 Channel n Compare Register 1 (ETIMER_CH*n*_COMP1)

This read/write register stores the value used for comparison with the counter value. This register is not byte accessible. More explanation on the use of COMP1 can be found in Usage of Compare Registers .

Address: 0h base + 0h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	COMP1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_COMP1 field descriptions

Field	Description
0–15 COMP1	Value used for comparison with the counter value

39.3.2 Channel n Compare Register 2 (ETIMER_CH*n*_COMP2)

This read/write register stores the value used for comparison with the counter value. This register is not byte accessible. More explanation on the use of COMP2 can be found in Usage of Compare Registers .

Address: 0h base + 2h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	COMP2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_COMP2 field descriptions

Field	Description
0–15 COMP2	Value used for comparison with the counter value

39.3.3 Channel n Capture Register 1 (ETIMER_CH*n*_CAPT1)

This read-only register stores the value captured from the counter. It is actually a 2-deep FIFO and not a single register. This register is not byte accessible. CCCTRL[CPT1MODE] determines when a capture occurs.

Address: 0h base + 4h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CAPT1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CAPT1 field descriptions

Field	Description
0–15 CAPT1	Value captured from the counter

39.3.4 Channel n Capture Register 2 (ETIMER_CH*n*_CAPT2)

This read-only register stores the value captured from the counter. It is actually a 2-deep FIFO and not a single register. This register is not byte accessible. CCCTRL[CPT2MODE] determines when a capture occurs.

Address: 0h base + 6h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CAPT2															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CAPT2 field descriptions

Field	Description
0–15 CAPT2	Value captured from the counter

39.3.5 Channel n Load Register (ETIMER_CH*n*_LOAD)

This read/write register stores the value used to initialize the counter. This register is not byte accessible.

Address: 0h base + 8h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	LOAD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_LOAD field descriptions

Field	Description
0–15 LOAD	Value used to initialize the counter

39.3.6 Channel n Hold Register (ETIMER_CH*n*_HOLD)

This read-only register stores the counter's value whenever any of the counters within a module are read. This is used to support coherent reading of cascaded counters.

Address: 0h base + Ah offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	HOLD															
Write	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_HOLD field descriptions

Field	Description
0–15 HOLD	Stores the counter's value whenever any of the counters within a module are read.

39.3.7 Channel n Counter Register (ETIMER_CH*n*_CNTR)

This read/write register is the counter for this channel of the timer module. This register is not byte accessible.

Address: 0h base + Ch offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CNTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CNTR field descriptions

Field	Description
0–15 CNTR	Counter for this channel of the timer module

39.3.8 Channel n Control Register 1 (ETIMER_CH*n*_CTRL1)

Address: 0h base + Eh offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7
Read Write	CNTMODE			PRISRC				
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	ONCE	LENGTH	DIR	SECSRC				
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CTRL1 field descriptions

Field	Description
0–2 CNTMODE	Count mode These bits control the basic counting and behavior of the counter. 000 No Operation 001 Count rising edges of primary source (Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1. If primary count source is IP bus clock, only rising edges are counted regardless of PIPS value.) 010 Count rising and falling edges of primary source (IP Bus clock divide by 1 can not be used as a primary count source in edge count mode.) 011 Count rising edges of primary source while secondary input high active 100 Quadrature count mode, uses primary and secondary sources 101 Count primary source rising edges, secondary source specifies direction (1 = minus) (Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1.)

Table continues on the next page...

ETIMER_CH*n*_CTRL1 field descriptions (continued)

Field	Description
	110 Edge of secondary source triggers primary count till compare 111 Cascaded counter mode, up/down (Primary count source must be set to one of the counter outputs.)
3–7 PRISRC	Primary Count Source These bits select the primary count source. **NOTE:** A timer selecting its own output as its primary count source is not a legal choice. The result is no counting. — 00000 Counter #0 input pin 00001 Counter #1 input pin 00010 Counter #2 input pin 00011 Counter #3 input pin 00100 Counter #4 input pin 00101 Counter #5 input pin 00110 Reserved 00111 Reserved 01000 Auxiliary input #0 pin 01001 Auxiliary input #1 pin 01010 Auxiliary input #2 pin 01011 Auxiliary input #3 pin 01100 Auxiliary input #4 pin 01101 Reserved 01110 Reserved 01111 Reserved 10000 Counter #0 output 10001 Counter #1 output 10010 Counter #2 output 10011 Counter #3 output 10100 Counter #4 output 10101 Counter #5 output 10110 Reserved 10111 Reserved 11000 IP Bus clock divide by 1 prescaler 11001 IP Bus clock divide by 2 prescaler 11010 IP Bus clock divide by 4 prescaler 11011 IP Bus clock divide by 8 prescaler 11100 IP Bus clock divide by 16 prescaler 11101 IP Bus clock divide by 32 prescaler 11110 IP Bus clock divide by 64 prescaler 11111 IP Bus clock divide by 128 prescaler
8 ONCE	Count once This bit selects continuous or one shot counting mode.

Table continues on the next page...

ETIMER_CH*n*_CTRL1 field descriptions (continued)

Field	Description
	0 Count repeatedly. 1 Count until compare and then stop. When output mode 4h is used, the counter re-initializes after reaching the COMP1 value and continues to count to the COMP2 value then stops.
9 LENGTH	Count Length This bit determines whether the counter counts to the compare value and then re-initializes itself to the value specified in the LOAD, CMPLD1, or CMPLD2 registers, or the counter continues counting past the compare value, to the binary roll over. 0 Continue counting to roll over. 1 Count until compare, then reinitialize. The value that the counter is reinitialized with depends on the settings of CLC1 and CLC2. If neither of these indicates the counter is to be loaded from one of the CMPLD registers, then the LOAD register is used to reinitialize the counter upon matching either COMP register. If one of CLC1 or CLC2 indicates that the counter is to be loaded from one of the CMPLD registers, then the counter will reinitialize to the value in the appropriate CMPLD register upon a match with the appropriate COMP register. If both of the CLC1 and CLC2 fields indicate that the counter is to be loaded from the CMPLD registers, then CMPLD1 will have priority if both compares happen at the same value. When output mode 4h is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until COMP1 value is reached, re-initializes, then counts until COMP2 value is reached, re-initializes, then counts until COMP1 value is reached, etc.
10 DIR	Count Direction This bit selects either the normal count direction up, or the reverse direction, down. 0 Count up 1 Count down
11–15 SECSRC	Secondary Count Source These bits identify the source to be used as a count command or timer command. The selected input can trigger the timer to capture the current value of the CNTR register. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the SIPS bit of the CTRL2 register. 00000 Counter #0 input pin 00001 Counter #1 input pin 00010 Counter #2 input pin 00011 Counter #3 input pin 00100 Counter #4 input pin 00101 Counter #5 input pin 00110 Reserved 00111 Reserved 01000 Auxiliary input #0 pin 01001 Auxiliary input #1 pin 01010 Auxiliary input #2 pin 01011 Auxiliary input #3 pin 01100 Auxiliary input #4 pin 01101 Reserved 01110 Reserved 01111 Reserved 10000 Counter #0 output 10001 Counter #1 output

Table continues on the next page...

ETIMER_CH*n*_CTRL1 field descriptions (continued)

Field	Description	
	10010	Counter #2 output
	10011	Counter #3 output
	10100	Counter #4 output
	10101	Counter #5 output
	10110	Reserved
	10111	Reserved
	11000-11111	Reserved

39.3.9 Channel n Control Register 2 (ETIMER_CH*n*_CTRL2)

Address: 0h base + 10h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7
Read	OEN	RDNT	INPUT	VAL	0	COFRC	COINIT	
Write			▨		FORCE			
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	SIPS	PIPS	OPS	MSTR	OUTMODE			
Write								
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CTRL2 field descriptions

Field	Description
0 OEN	Output Enable This bit determines the direction of the external pin. 0 The external pin is configured as an input. 1 OFLAG output signal will be driven on the external pin. Other timer channels using this external pin as their input will see the driven value. The polarity of the signal will be determined by the OPS bit.
1 RDNT	Redundant Channel Enable This bit enables redundant channel checking between adjacent channels (0 and 1, 2 and 3, 4 and 5). When this bit is clear, the RCF bit in this channel cannot be set. When this bit is set, the RCF bit will be set by a miscompare between the OFLAG of this channel and the OFLAG of its redundant adjacent channel which will cause the output of this channel to go inactive (logic 0 prior to consideration of the OPS bit). 0 Disable redundant channel checking. 1 Enable redundant channel checking.
2 INPUT	External input signal This read only bit reflects the current state of the signal selected via SECSRC after application of the SIPS bit and filtering.
3 VAL	Forced OFLAG Value

Table continues on the next page...

ETIMER_CH*n*_CTRL2 field descriptions (continued)

Field	Description
	This bit determines the value of the OFLAG output signal when a software triggered FORCE command occurs.
4 FORCE	Force the OFLAG output This write only bit forces the current value of the VAL bit to be written to the OFLAG output. This bit always reads as a zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if OUTMODE is 0000 (software controlled). Setting this bit while the OUTMODE is a different value may yield unpredictable results.
5 COFRC	Co-channel OFLAG Force This bit enables the compare from another channel within the module to force the state of this counter's OFLAG output signal. 0 Other channels cannot force the OFLAG of this channel. 1 Other channels may force the OFLAG of this channel.
6–7 COINIT	Co-channel Initialization These bits enable another channel within the module to force the re-initialization of this channel when the other channel has an active compare event. 00 Other channels cannot force re-initialization of this channel. 01 Other channels may force a re-initialization of this channel's counter using the LOAD reg. 10 Other channels may force a re-initialization of this channel's counter with the CMPLD2 reg when this channel is counting down or the CMPLD1 reg when this channel is counting up. 11 Reserved
8 SIPS	Secondary Source Input Polarity Select This bit inverts the polarity of the signal selected by the SECSRC bits. 0 True polarity. 1 Inverted polarity.
9 PIPS	Primary Source Input Polarity Select This bit inverts the polarity of the signal selected by the PRISRC bits. This only applies if the signal selected by PRISRC is not the prescaled IP Bus clock. 0 True polarity. 1 Inverted polarity.
10 OPS	Output Polarity Select. This bit inverts the OFLAG output signal polarity. 0 True polarity. 1 Inverted polarity.
11 MSTR	Master Mode This bit enables the compare function's output to be broadcasted to the other channels in the module. The compare signal then can be used to reinitialize the other counters and/or force their OFLAG signal outputs. 0 Disable broadcast of compare events from this channel. 1 Enable broadcast of compare events from this channel.

Table continues on the next page...

ETIMER_CH*n*_CTRL2 field descriptions (continued)

Field	Description
12–15 OUTMODE	Output Mode These bits determine the mode of operation for the OFLAG output signal. 0000 Software controlled 0001 Clear OFLAG output on successful compare (COMP1 or COMP2) 0010 Set OFLAG output on successful compare (COMP1 or COMP2) 0011 Toggle OFLAG output on successful compare (COMP1 or COMP2) 0100 Toggle OFLAG output using alternating compare registers 0101 Set on compare with COMP1, cleared on secondary source input edge 0110 Set on compare with COMP2, cleared on secondary source input edge 0111 Set on compare, cleared on counter roll-over 1000 Set on successful compare on COMP1, clear on successful compare on COMP2 1001 Asserted while counter is active, cleared when counter is stopped. 1010 Asserted when counting up, cleared when counting down. 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Enable gated clock output while counter is active

39.3.10 Channel n Control Register 3 (ETIMER_CH*n*_CTRL3)

Address: 0h base + 12h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7
Read Write	STPEN	ROC		Reserved				
Reset	0	0	0	0	1	1	1	1

Bit	8	9	10	11	12	13	14	15
Read	C2FCNT			C1FCNT			DBGEN	
Write	▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CTRL3 field descriptions

Field	Description
0 STPEN	Stop Actions Enable This bit allows the tri-stating of the timer output during stop mode. 0 Output enable is unaffected by stop mode. 1 Output enable is disabled during stop mode.
1–2 ROC	Reload on Capture

Table continues on the next page...

ETIMER_CH*n*_CTRL3 field descriptions (continued)

Field	Description
	These bits enable the capture function to cause the counter to be reloaded from the LOAD register. 00 Do not reload the counter on a capture event. 01 Reload the counter on a capture 1 event. 10 Reload the counter on a capture 2 event. 11 Reload the counter on both a capture 1 event and a capture 2 event.
3–7 Reserved	This field is reserved.
8–10 C2FCNT	Capture 2 FIFO count This field reflects the number of words in the CAPT2 FIFO.
11–13 C1FCNT	Capture 1 FIFO count This field reflects the number of words in the CAPT1 FIFO.
14–15 DBGEN	Debug Actions Enable These bits allow the counter channel to perform certain actions in response to the chip entering debug mode. 00 Continue with normal operation during debug mode. (default) 01 Halt channel counter during debug mode. 10 Force OFLAG to logic 0 (prior to consideration of the OPS bit) during debug mode. 11 Both halt counter and force OFLAG to 0 during debug mode.

39.3.11 Channel n Status Register (ETIMER_CH*n*_STS)

Address: 0h base + 14h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						WDF	RCF	ICF2	ICF1	IEHF	IELF	TOF	TCF2	TCF1	TCF
Write							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_STS field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 WDF	Watchdog Time-out Flag This bit is set when the watchdog times out by counting down to zero. The watchdog must be enabled for time-out to occur and channel 0 must be in quadrature decode count mode (CNTMODE = 100). This bit is cleared by writing a 1 to this bit position after either writing a non-zero value to WDTOL and/or WDTOH or exiting quadrature decode counting mode. This bit is only in channel 0. If the watchdog timer is not implemented, then this bit is read-only zero.
7 RCF	Redundant Channel Flag

Table continues on the next page...

ETIMER_CHn_STS field descriptions (continued)

Field	Description
	This field is set to 1 when there is a miscompare between this channel's OFLAG value and the OFLAG value of the corresponding redundant channel. Corresponding channels are grouped together in the following pairs: 0 and 1, 2 and 3, 4 and 5 . This field can only be set to 1 if the RDNT bit is set. This field is cleared by writing a 1 to this field. **NOTE:** This field is functional only in the even channels, i.e., 0, 2, 4 . In the odd channels, this field cannot create an interrupt.
8 ICF2	Input Capture 2 Flag This bit is set when an input capture event (as defined by CPT2MODE) occurs and the word count of the CAPT2 FIFO exceeds the value of the CFWM field. This bit is cleared by writing a one to this bit position if ICF2DE is clear (no DMA) or it is cleared automatically by the DMA access if ICF2DE is set (DMA).
9 ICF1	Input Capture 1 Flag This bit is set when an input capture event (as defined by CPT1MODE) occurs and the word count of the CAPT1 FIFO exceeds the value of the CFWM field. This bit is cleared by writing a one to this bit position if ICF1DE is clear (no DMA) or it is cleared automatically by the DMA access if ICF1DE is set (DMA).
10 IEHF	Input Edge High Flag This bit is set when a positive input transition occurs (on an input selected by SECSRC) while the counter is enabled. This bit is cleared by writing a one to this bit position.
11 IELF	Input Edge Low Flag This bit is set when a negative input transition occurs (on an input selected by SECSRC) while the counter is enabled. This bit is cleared by writing a one to this bit position.
12 TOF	Timer Overflow Flag This bit is set when the counter rolls over its maximum value FFFFh or 0000h (depending on count direction). This bit is cleared by writing a one to this bit location.
13 TCF2	Timer Compare 2 Flag This bit is set when a successful compare occurs with COMP2. This bit is cleared by writing a one to this bit location.
14 TCF1	Timer Compare 1 Flag This bit is set when a successful compare occurs with COMP1. This bit is cleared by writing a one to this bit location.
15 TCF	This bit is set when a successful compare occurs. This bit is cleared by writing a one to this bit location.

39.3.12 Channel n Interrupt and DMA Enable Register (ETIMER_CH*n*_INTDMA)

Address: 0h base + 16h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7
Read	ICF2DE	ICF1DE	CMPLD2DE	CMPLD1DE	0		WDFIE	RCFIE
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	ICF2IE	ICF1IE	IEHFIE	IELFIE	TOFIE	TCF2IE	TCF1IE	TCFIE
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_INTDMA field descriptions

Field	Description
0 ICF2DE	Input Capture 2 Flag DMA Enable Setting this bit enables DMA read requests for CAPT2 when the ICF2 bit is set. Do not set both this bit and the ICF2IE bit.
1 ICF1DE	Input Capture 1 Flag DMA Enable Setting this bit enables DMA read requests for CAPT1 when the ICF1 bit is set. Do not set both this bit and the ICF1IE bit.
2 CMPLD2DE	Comparator Load Register 2 Flag DMA Enable Setting this bit enables DMA write requests to the CMPLD2 register whenever data is transferred out of the CMPLD2 reg into either the CNTR, COMP1, or COMP2 registers.
3 CMPLD1DE	Comparator Load Register 1 Flag DMA Enable Setting this bit enables DMA write requests to the CMPLD1 register whenever data is transferred out of the CMPLD1 reg into either the CNTR, COMP1, or COMP2 registers.
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 WDFIE	Watchdog Flag Interrupt Enable Setting this bit enables interrupts when the WDF bit is set. This bit is only in channel 0. If the watchdog timer is not implemented, then this bit serves no purpose.
7 RCFIE	Redundant Channel Flag Interrupt Enable Setting this bit enables interrupts when the RCF bit is set. This bit is only in even channels (0, 2, and 4).
8 ICF2IE	Input Capture 2 Flag Interrupt Enable Setting this bit enables interrupts when the ICF2 bit is set. Do not set both this bit and the ICF2DE bit.
9 ICF1IE	Input Capture 1 Flag Interrupt Enable Setting this bit enables interrupts when the ICF1 bit is set. Do not set both this bit and the ICF1DE bit.
10 IEHFIE	Input Edge High Flag Interrupt Enable Setting this bit enables interrupts when the IEHF bit is set.

Table continues on the next page...

ETIMER_CH*n*_INTDMA field descriptions (continued)

Field	Description
11 IELFIE	Input Edge Low Flag Interrupt Enable Setting this bit enables interrupts when the IELF bit is set.
12 TOFIE	Timer Overflow Flag Interrupt Enable Setting this bit enables interrupts when the TOF bit is set.
13 TCF2IE	Timer Compare 2 Flag Interrupt Enable Setting this bit enables interrupts when the TCF2 bit is set.
14 TCF1IE	Timer Compare 1 Flag Interrupt Enable Setting this bit enables interrupts when the TCF1 bit is set.
15 TCFIE	Timer Compare Flag Interrupt Enable Setting this bit enables interrupts when the TCF bit is set.

39.3.13 Channel n Comparator Load Register 1 (ETIMER_CH*n*_CMPLD1)

This read/write register is the preload value for the COMP1 register. This register can also be used to load into the CNTR register. This register is not byte accessible. More information on the use of this register can be found in Usage of Compare Load Registers .

Address: 0h base + 18h offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CMPLD1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CMPLD1 field descriptions

Field	Description
0–15 CMPLD1	Preload value for the COMP1 register

39.3.14 Channel n Comparator Load Register 2 (ETIMER_CH*n*_CMPLD2)

This read/write register is the preload value for the COMP2 register. This register can also be used to load into the CNTR register. This register is not byte accessible. More information on the use of this register can be found in Usage of Compare Load Registers

Address: 0h base + 1Ah offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CMPLD2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CMPLD2 field descriptions

Field	Description
0–15 CMPLD2	Preload value for the COMP2 register

39.3.15 Channel n Compare and Capture Control Register (ETIMER_CH*n*_CCCTRL)

Address: 0h base + 1Ch offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7
Read Write	CLC2			CLC1			CMPMODE	
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CPT2MODE		CPT1MODE		CFWM		ONESHOT	ARM
Reset	0	0	0	0	0	0	0	0

ETIMER_CH*n*_CCCTRL field descriptions

Field	Description
0–2 CLC2	Compare Load Control 2 These bits control when COMP2 is preloaded. It also controls the loading of CNTR. 000 Never preload 001 Reserved 010 Load COMP2 with CMPLD1 upon successful compare with the value in COMP1. 011 Load COMP2 with CMPLD1 upon successful compare with the value in COMP2. 100 Load COMP2 with CMPLD2 upon successful compare with the value in COMP1. 101 Load COMP2 with CMPLD2 upon successful compare with the value in COMP2.

Table continues on the next page...

ETIMER_CH*n*_CCCTRL field descriptions (continued)

Field	Description
	110 Load CNTR with CMPLD2 upon successful compare with the value in COMP1. 111 Load CNTR with CMPLD2 upon successful compare with the value in COMP2.
3–5 CLC1	Compare Load Control 1 These bits control when COMP1 is preloaded. It also controls the loading of CNTR. 000 Never preload 001 Reserved 010 Load COMP1 with CMPLD1 upon successful compare with the value in COMP1. 011 Load COMP1 with CMPLD1 upon successful compare with the value in COMP2. 100 Load COMP1 with CMPLD2 upon successful compare with the value in COMP1. 101 Load COMP1 with CMPLD2 upon successful compare with the value in COMP2. 110 Load CNTR with CMPLD1 upon successful compare with the value in COMP1. 111 Load CNTR with CMPLD1 upon successful compare with the value in COMP2.
6–7 CMPMODE	Compare Mode These bits control when the COMP1 and COMP2 registers are used in regards to the counting direction. 00 COMP1 register is used when the counter is counting up; COMP2 register is used when the counter is counting up. 01 COMP1 register is used when the counter is counting down; COMP2 register is used when the counter is counting up. 10 COMP1 register is used when the counter is counting up. COMP2 register is used when the counter is counting down. 11 COMP1 register is used when the counter is counting down; COMP2 register is used when the counter is counting down.
8–9 CPT2MODE	Capture 2 Mode Control These bits control the operation of the CAPT2 register as well as the operation of the ICF2 flag by defining which input edges cause a capture event. The input source is the secondary count source. 00 Disabled 01 Capture falling edges 10 Capture rising edges. 11 Capture any edge.
10–11 CPT1MODE	Capture 1 Mode Control These bits control the operation of the CAPT1 register as well as the operation of the ICF1 flag by defining which input edges cause a capture event. The input source is the secondary count source. 00 Disabled 01 Capture falling edges 10 Capture rising edges. 11 Capture any edge.
12–13 CFWM	Capture FIFO Water Mark This field represents the water mark level for the CAPT1 and CAPT2 FIFOs. The capture flags, ICF1 and ICF2, won't be set until the word count of the corresponding FIFO is equal to or greater than this water mark level.
14 ONESHOT	One Shot Capture Mode This bit selects between free running and one shot mode for the input capture circuitry.

Table continues on the next page...

ETIMER_CH*n*_CCCTRL field descriptions (continued)

Field	Description
	0 Free running mode is selected. If both capture circuits are enabled, then capture circuit 1 is armed first after the ARM bit is set. Once a capture occurs, capture circuit 1 is disarmed and capture circuit 2 is armed. After capture circuit 2 performs a capture, it is disarmed and capture circuit 1 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 1 is armed first after the ARM bit is set. Once a capture occurs, capture circuit 1 is disarmed and capture circuit 2 is armed. After capture circuit 2 performs a capture, it is disarmed and the ARM bit is cleared. No further captures will be performed until the ARM bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARM bit is then cleared.
15 ARM	Arm Capture Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s). 0 Input capture operation is disabled. 1 Input capture operation as specified by the CPT1MODE and CPT2MODE bits is enabled.

39.3.16 Channel n Input Filter Register (ETIMER_CH*n*_FILT)

The FILT register programs the values for the filtering of the corresponding input without regard for the fact that any eTimer channel can use the input as a count source. The FILT register is also used to control the filtering of the corresponding auxilliary input if one exists.

The FILT_PER value should be set such that the sampling period is larger the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT + 3 power.

The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of: (((FILT_CNT + 3) x FILT_PER) + 2) peripheral clock periods.

Address: 0h base + 1Eh offset + (32d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FILT_CNT			FILT_PER							
Write	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_CH*n*_FILT field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 FILT_CNT	Input Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FILT_CNT affects the input latency as described in this section.
8–15 FILT_PER	Input Filter Sample Period These bits represent the sampling period (in peripheral clock cycles) of the eTimer input signal. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is $00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency as described in in this section.

39.3.17 Watchdog Time-out Low Word Register (ETIMER_WDTOL)

The fields in WDTOL and WDTOH are combined to form the 32 bit time-out count, WDTO, for the Timer watchdog function. This time-out count is used to monitor for inactivity on the inputs when channel 0 is in the quadrature decode count mode. The watchdog function is enabled whenever WDTO contains a non-zero value (although actual counting only occurs if channel 0 is in quadrature decode counting mode). The watchdog time-out down counter is loaded whenever WDTOH is written. This register is not byte accessible. See Watchdog Timer for more information on the use of the watchdog timer.

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	WDTOL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_WDTOL field descriptions

Field	Description
0–15 WDTOL	Low word of the 32-bit time-out count (WDTO) for the Timer watchdog function

39.3.18 Watchdog Time-out High Word Register (ETIMER_WDTOH)

The fields in WDTOL and WDTOH are combined to form the 32 bit time-out count, WDTO, for the Timer watchdog function. This time-out count is used to monitor for inactivity on the inputs when channel 0 is in the quadrature decode count mode. The watchdog function is enabled whenever WDTO contains a non-zero value (although actual counting only occurs if channel 0 is in quadrature decode counting mode). The watchdog time-out down counter is loaded whenever WDTOH is written. This register is not byte accessible. See Watchdog Timer for more information on the use of the watchdog timer.

Address: 0h base + 102h offset = 102h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	WDTOH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ETIMER_WDTOH field descriptions

Field	Description
0–15 WDTOH	High word of the 32-bit time-out count (WDTO) for the Timer watchdog function

39.3.19 Channel Enable Register (ETIMER_ENBL)

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0										ENBL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

ETIMER_ENBL field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 ENBL	Timer Channel Enable These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate channels. If an ENBL bit is set, then the corresponding channel will start counting as soon as the CNTMODE field has a value other than 000. When an ENBL bit is clear, the corresponding channel maintains its current value. Bit value 0 = Timer channel is disabled.

Table continues on the next page...

ETIMER_ENBL field descriptions (continued)

Field	Description
	Bit value 1 = Timer channel is enabled. (default)

39.3.20 DMA Request 0 Select Register (ETIMER_DREQ0)

Address: 0h base + 110h offset = 110h

Bit	0	1	2	3	4	5	6	7
Read	DREQ0_EN	0						
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			DREQ0				
Write								
Reset	0	0	0	0	0	0	0	0

ETIMER_DREQ0 field descriptions

Field	Description
0 DREQ0_EN	DMA Request Enable Use this bit to enable the module-level DMA request output. Program the DREQ field prior to setting the enable bit. Clearing this enable bit will remove the request but won't clear the flag that is causing the request. 0 DMA request disabled. 1 DMA request enabled.
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DREQ0	DMA Request Select Use this field to select which DMA request source will be muxed onto one of the four module level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed. 00000: Channel 0 CAPT1 DMA read request 00001: Channel 0 CAPT2 DMA read request 00010: Channel 0 CMPLD1 DMA write request 00011: Channel 0 CMPLD2 DMA write request 00100: Channel 1 CAPT1 DMA read request 00101: Channel 1 CAPT2 DMA read request 00110: Channel 1 CMPLD1 DMA write request 00111: Channel 1 CMPLD2 DMA write request 01000: Channel 2 CAPT1 DMA read request 01001: Channel 2 CAPT2 DMA read request

Table continues on the next page...

ETIMER_DREQ0 field descriptions (continued)

Field	Description
	01010: Channel 2 CMPLD1 DMA write request 01011: Channel 2 CMPLD2 DMA write request 01100: Channel 3 CAPT1 DMA read request 01101: Channel 3 CAPT2 DMA read request 01110: Channel 3 CMPLD1 DMA write request 01111: Channel 3 CMPLD2 DMA write request 10000: Channel 4 CAPT1 DMA read request 10001: Channel 4 CAPT2 DMA read request 10010: Channel 4 CMPLD1 DMA write request 10011: Channel 4 CMPLD2 DMA write request 10100: Channel 5 CAPT1 DMA read request 10101: Channel 5 CAPT2 DMA read request 10110: Channel 5 CMPLD1 DMA write request 10111: Channel 5 CMPLD2 DMA write request 11000: Reserved 11001: Reserved 11010: Reserved 11011: Reserved 11100: Reserved 11101: Reserved 11110: Reserved 11111: Reserved

39.3.21 DMA Request 1 Select Register (ETIMER_DREQ1)

Address: 0h base + 112h offset = 112h

Bit	0	1	2	3	4	5	6	7
Read	DREQ1_EN	0						
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			DREQ1				
Write								
Reset	0	0	0	0	0	0	0	0

ETIMER_DREQ1 field descriptions

Field	Description
0 DREQ1_EN	DMA Request Enable Use this bit to enable the module-level DMA request output. Program the DREQ field prior to setting the enable bit. Clearing this enable bit will remove the request but won't clear the flag that is causing the request. 0 DMA request disabled. 1 DMA request enabled.
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DREQ1	DMA Request Select Use this field to select which DMA request source will be muxed onto one of the four module level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed. 00000: Channel 0 CAPT1 DMA read request 00001: Channel 0 CAPT2 DMA read request 00010: Channel 0 CMPLD1 DMA write request 00011: Channel 0 CMPLD2 DMA write request 00100: Channel 1 CAPT1 DMA read request 00101: Channel 1 CAPT2 DMA read request 00110: Channel 1 CMPLD1 DMA write request 00111: Channel 1 CMPLD2 DMA write request 01000: Channel 2 CAPT1 DMA read request 01001: Channel 2 CAPT2 DMA read request 01010: Channel 2 CMPLD1 DMA write request 01011: Channel 2 CMPLD2 DMA write request 01100: Channel 3 CAPT1 DMA read request 01101: Channel 3 CAPT2 DMA read request 01110: Channel 3 CMPLD1 DMA write request 01111: Channel 3 CMPLD2 DMA write request 10000: Channel 4 CAPT1 DMA read request 10001: Channel 4 CAPT2 DMA read request 10010: Channel 4 CMPLD1 DMA write request 10011: Channel 4 CMPLD2 DMA write request 10100: Channel 5 CAPT1 DMA read request 10101: Channel 5 CAPT2 DMA read request 10110: Channel 5 CMPLD1 DMA write request 10111: Channel 5 CMPLD2 DMA write request 11000: Reserved 11001: Reserved

Table continues on the next page...

ETIMER_DREQ1 field descriptions (continued)

Field	Description
	11010: Reserved 11011: Reserved 11100: Reserved 11101: Reserved 11110: Reserved 11111: Reserved

39.3.22 DMA Request 2 Select Register (ETIMER_DREQ2)

NOTE

DREQ2 is implemented only in eTimer instantiations that support three or more DMA request outputs.

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7
Read	DREQ2_EN	0						
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			DREQ2				
Write								
Reset	0	0	0	0	0	0	0	0

ETIMER_DREQ2 field descriptions

Field	Description
0 DREQ2_EN	DMA Request Enable Use this bit to enable the module-level DMA request output. Program the DREQ field prior to setting the enable bit. Clearing this enable bit will remove the request but won't clear the flag that is causing the request. 0 DMA request disabled. 1 DMA request enabled.
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DREQ2	DMA Request Select Use this field to select which DMA request source will be muxed onto one of the four module level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed.

Table continues on the next page...

ETIMER_DREQ2 field descriptions (continued)

Field	Description
	00000: Channel 0 CAPT1 DMA read request 00001: Channel 0 CAPT2 DMA read request 00010: Channel 0 CMPLD1 DMA write request 00011: Channel 0 CMPLD2 DMA write request 00100: Channel 1 CAPT1 DMA read request 00101: Channel 1 CAPT2 DMA read request 00110: Channel 1 CMPLD1 DMA write request 00111: Channel 1 CMPLD2 DMA write request 01000: Channel 2 CAPT1 DMA read request 01001: Channel 2 CAPT2 DMA read request 01010: Channel 2 CMPLD1 DMA write request 01011: Channel 2 CMPLD2 DMA write request 01100: Channel 3 CAPT1 DMA read request 01101: Channel 3 CAPT2 DMA read request 01110: Channel 3 CMPLD1 DMA write request 01111: Channel 3 CMPLD2 DMA write request 10000: Channel 4 CAPT1 DMA read request 10001: Channel 4 CAPT2 DMA read request 10010: Channel 4 CMPLD1 DMA write request 10011: Channel 4 CMPLD2 DMA write request 10100: Channel 5 CAPT1 DMA read request 10101: Channel 5 CAPT2 DMA read request 10110: Channel 5 CMPLD1 DMA write request 10111: Channel 5 CMPLD2 DMA write request 11000: Reserved 11001: Reserved 11010: Reserved 11011: Reserved 11100: Reserved 11101: Reserved 11110: Reserved 11111: Reserved

39.3.23 DMA Request 3 Select Register (ETIMER_DREQ3)

NOTE

DREQ3 is implemented only in eTimer instantiations that support four or more DMA request outputs.

Address: 0h base + 116h offset = 116h

Bit	0	1	2	3	4	5	6	7
Read	DREQ3_EN	0						
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			DREQ3				
Write								
Reset	0	0	0	0	0	0	0	0

ETIMER_DREQ3 field descriptions

Field	Description
0 DREQ3_EN	DMA Request Enable Use this bit to enable the module-level DMA request output. Program the DREQ field prior to setting the enable bit. Clearing this enable bit will remove the request but won't clear the flag that is causing the request. 0 DMA request disabled. 1 DMA request enabled.
1–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DREQ3	DMA Request Select Use this field to select which DMA request source will be muxed onto one of the four module level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed. 00000: Channel 0 CAPT1 DMA read request 00001: Channel 0 CAPT2 DMA read request 00010: Channel 0 CMPLD1 DMA write request 00011: Channel 0 CMPLD2 DMA write request 00100: Channel 1 CAPT1 DMA read request 00101: Channel 1 CAPT2 DMA read request 00110: Channel 1 CMPLD1 DMA write request 00111: Channel 1 CMPLD2 DMA write request 01000: Channel 2 CAPT1 DMA read request

Table continues on the next page...

ETIMER_DREQ3 field descriptions (continued)

Field	Description
	01001: Channel 2 CAPT2 DMA read request 01010: Channel 2 CMPLD1 DMA write request 01011: Channel 2 CMPLD2 DMA write request 01100: Channel 3 CAPT1 DMA read request 01101: Channel 3 CAPT2 DMA read request 01110: Channel 3 CMPLD1 DMA write request 01111: Channel 3 CMPLD2 DMA write request 10000: Channel 4 CAPT1 DMA read request 10001: Channel 4 CAPT2 DMA read request 10010: Channel 4 CMPLD1 DMA write request 10011: Channel 4 CMPLD2 DMA write request 10100: Channel 5 CAPT1 DMA read request 10101: Channel 5 CAPT2 DMA read request 10110: Channel 5 CMPLD1 DMA write request 10111: Channel 5 CMPLD2 DMA write request 11000: Reserved 11001: Reserved 11010: Reserved 11011: Reserved 11100: Reserved 11101: Reserved 11110: Reserved 11111: Reserved

39.4 Functional Description

This section provides a functional description of eTimer.

39.4.1 General

Each channel has two basic modes of operation: it can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.

- The counter can decode and count quadrature encoded input signals.
- The counter can count up and down using dual inputs in a "count with direction" format.
- The counter's terminal count value (modulo) is programmable.
    - The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count "rolls over" to zero.

The external inputs to each counter/timer are shareable among each of the channels within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be "captured"
- They can be used to generate interrupt requests

The polarity of the external inputs is selectable.

The primary output of each channel is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is programmable.

Any channel can be assigned as a "Master". A master's compare signal can be broadcasted to the other channels within the module. The other channels can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a Master channel's compare event occurs.

39.4.2 Counting Modes

The selected external signals are sampled at the eTimer's base clock rate and then run through a transition detector. The maximum count rate is one-half of the eTimer's base clock rate when using an external signal. Internal clock sources can be used to clock the counters at the eTimer's base clock rate.

If a counter is programmed to count to a specific value and then stop, the CNTMODE field in the CTRL1 register is cleared when the count terminates.

39.4.2.1 STOP Mode

If the CNTMODE field is set to '000', the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

39.4.2.2 COUNT Mode

If the CNTMODE field is set to '001', the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as "widgets" on a conveyor belt passing a sensor. If the selected input is inverted by setting the PIPS bit, then the negative edge of the selected external input signal is counted.

See CASCADE-COUNT Mode through VARIABLE-FREQUENCY PWM Mode for additional capabilities of this operating mode.

39.4.2.3 EDGE-COUNT Mode

If the CNTMODE field is set to '010', the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment such as a simple encoder wheel.

39.4.2.4 GATED-COUNT Mode

If the CNTMODE field is set to '011', the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting the SIPS bit, then the counter will count while the selected secondary input is low.

39.4.2.5 QUADRATURE-COUNT Mode

If the CNTMODE field is set to 100, the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signals provides both count and direction information.

The following figure shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

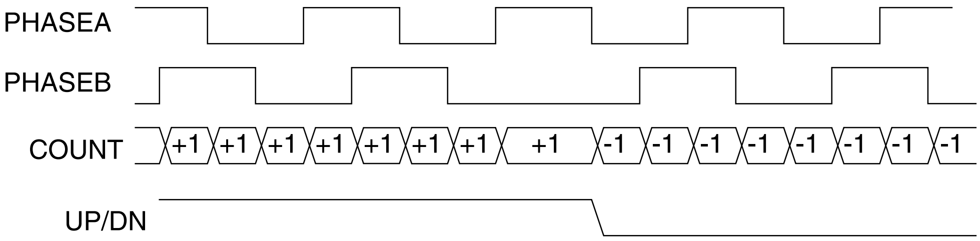


Figure 39-3. Quadrature Incremental Position Encoder

39.4.2.6 SIGNED-COUNT Mode

If the CNTMODE field is set to '101', the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

39.4.2.7 TRIGGERED-COUNT Mode

If the CNTMODE field is set to '110', the counter will begin counting the primary clock source after a positive transition (negative if SIPS = 1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. Subsequent secondary positive input transitions will continue to restart and stop the counting until a compare event occurs.

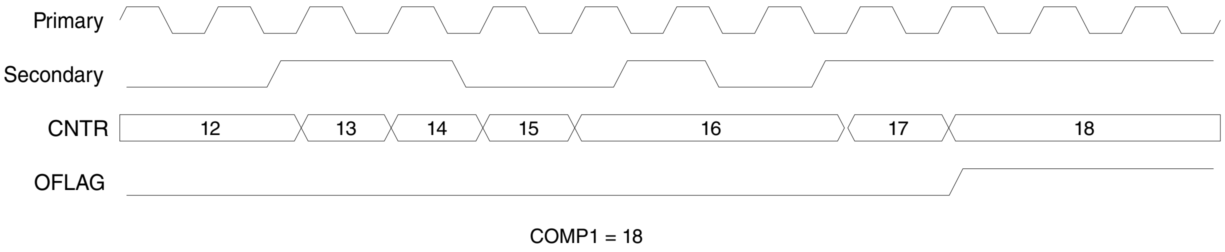


Figure 39-4. Triggered Count Mode (Length=1)

39.4.2.8 ONE-SHOT Mode

If the CNTMODE field is set to '110', and the counter is set to reinitialize at a compare event (LENGTH =1), and the OFLAG OUTMODE is set to '0101' (cleared on init, set on compare), the counter works in a "One-Shot Mode". An external events causes the counter to count, when terminal count is reached, the output is asserted. This "delayed" output can be used to provide timing delays.

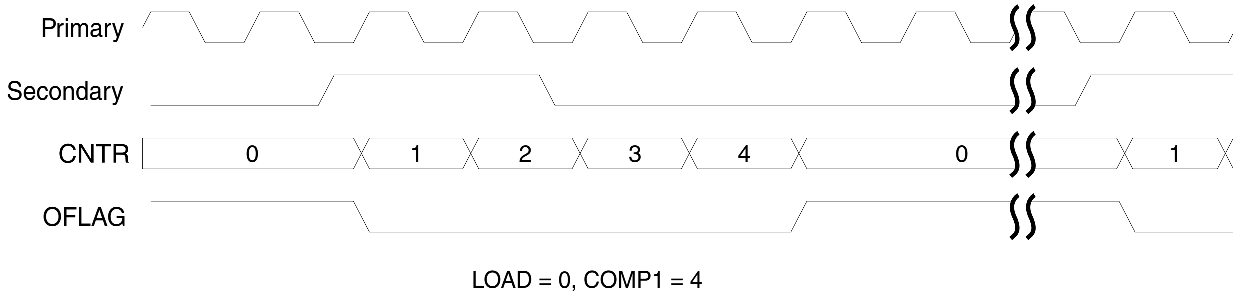


Figure 39-5. One-Shot Mode (Length=1)

39.4.2.9 CASCADE-COUNT Mode

If the CNTMODE field is set to '111', the counter's input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This "Cascade" or "Daisy-Chained" mode enables multiple counters to be cascaded to yield longer counter lengths. When operating in cascade mode, a special high speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented.

Up to two counters may be cascaded to create a 32 bit wide synchronous counter.

Whenever any counter is read within a counter module, all of the counters' values within the module are captured in their respective HOLD registers. This action supports the reading of a cascaded counter chain. First read any counter of a cascaded counter chain, then read the HOLD registers of the other counters in the chain. The cascaded counter mode is synchronous.

Note

It is possible to connect counters together by using the other (non-cascade) counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a "ripple" mode, where higher order counters will transition a clock later than a purely synchronous design.

Note

A channel can be cascaded with any other channel, but you can't cascade more than 2 channels together. You can create separate cascades of pairs of channels. For example, you can cascade channels 0 and 1 and separately cascade channels 6 and 5. You can't cascade channels 0, 1, and 5.

39.4.2.10 PULSE-OUTPUT Mode

If the counter is setup for CNTMODE = 001, and the OFLAG OUTMODE is set to '1111' (gated clock output), and the ONCE bit is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value. This mode is useful for driving step motor systems.

Note

This does not work if the PRISRC is set to 11000 (IP_bus/1).

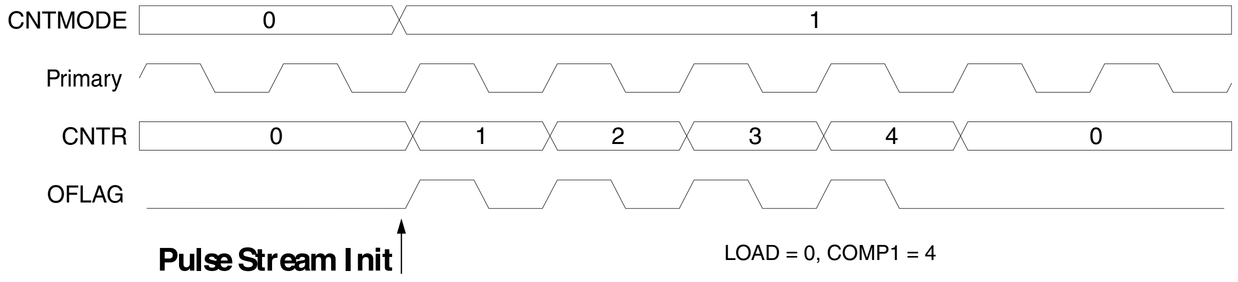


Figure 39-6. Pulse Output Mode

39.4.2.11 FIXED-FREQUENCY PWM Mode

If the counter is setup for CNTMODE = 001, count through roll-over (LENGTH = 0), continuous count (ONCE = 0) and the OFLAG OUTMODE is '0111' (set on compare, cleared on counter roll-over) then the counter output yields a Pulse Width Modulated

(PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

39.4.2.12 VARIABLE-FREQUENCY PWM Mode

If the counter is setup for CNTMODE = 001, count till compare (LENGTH = 1), continuous count (ONCE = 0) and the OFLAG OUTMMODE is '0100' (toggle OFLAG and alternate compare registers) then the counter output yields a Pulse Width Modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

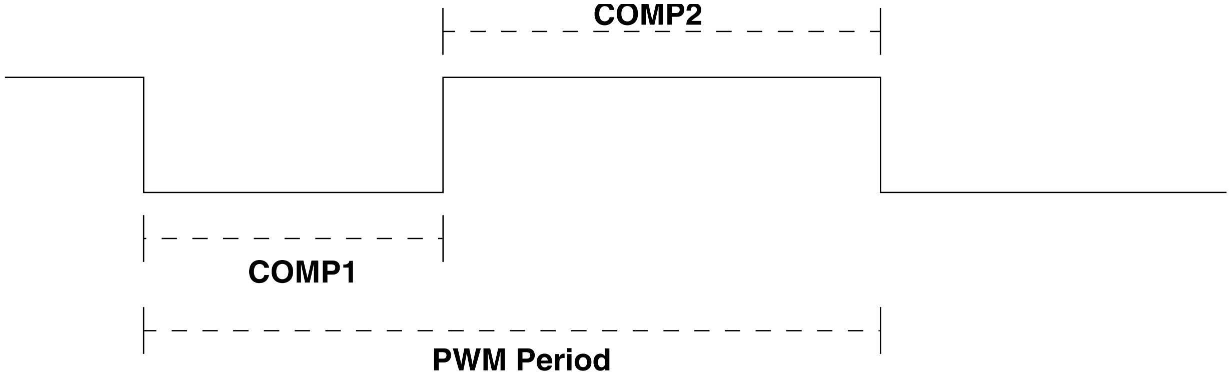
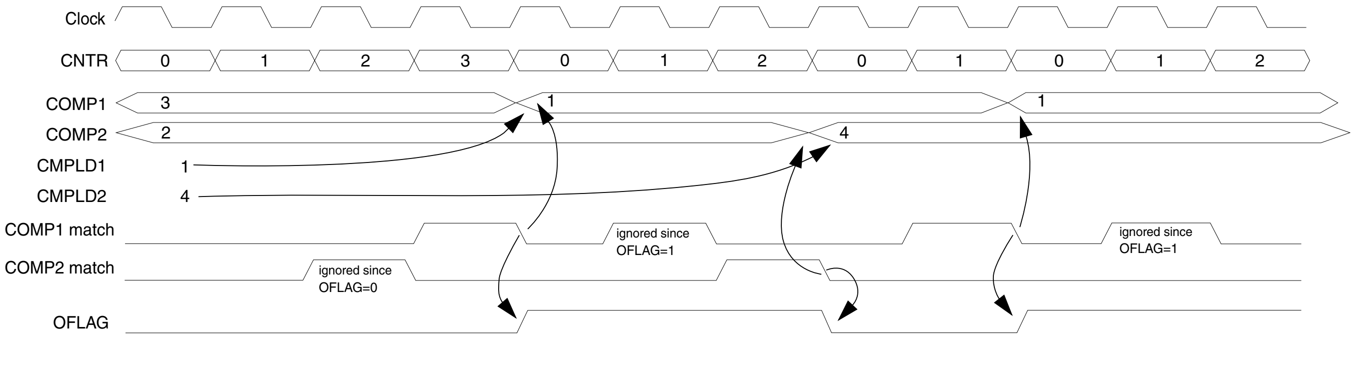


Figure 39-7. Variable PWM Waveform



CLC1=010, load COMP1 when CNTR=COMP1
CLC2=101, load COMP2 when CNTR=COMP2

Figure 39-8. Variable Frequency PWM Mode Timing

39.4.2.13 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability.

The COMP1 register should be set to the desired maximum count value or FFFFh to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or 0000h to indicate the minimum unsigned value prior to roll-under.

If the output mode is set to 0100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time. COMP1 is used when OFLAG=0 and COMP2 is used when OFLAG=1. OFLAG can be forced to a value using CTRL2[FORCE] and CTRL2[VAL].

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to FFFFh or 0000h, roll over, then begin counting toward the new value. The check is: CNTR = COMPx, *not* CNTR > COMP1 or CNTR < COMP2.

The use of the CMPLD1 and CMPLD2 registers to preload compare values will help to minimize this problem.

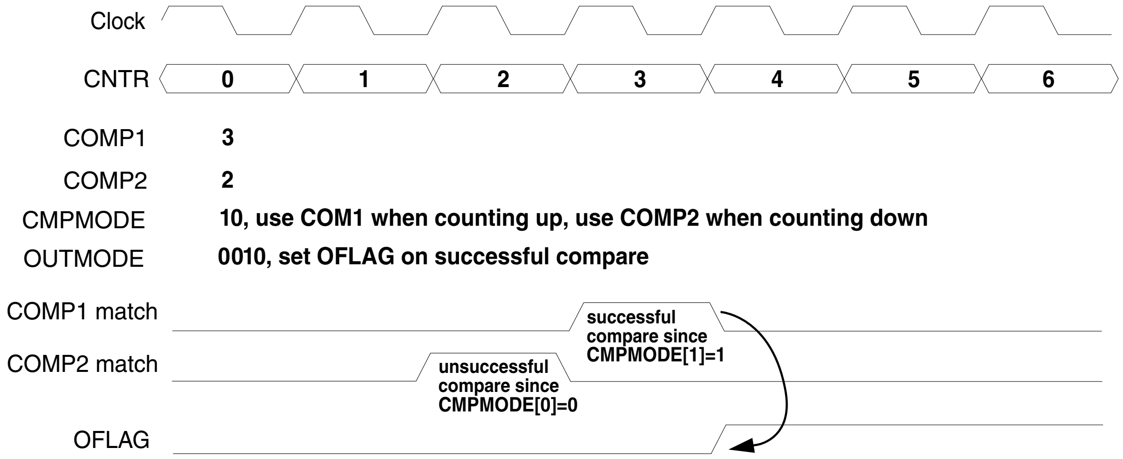

Clock
CNTR
0
1
2
3
4
5
6
COMP1
3
COMP2
2
CMPMODE
10, use COM1 when counting up, use COMP2 when counting down
OUTMODE
0010, set OFLAG on successful compare
COMP1 match
successful compare since CMPMODE[1]=1
COMP2 match
unsuccessful compare since CMPMODE[0]=0
OFLAG

Figure 39-9. Compare Register and OFLAG Timing

39.4.2.14 Usage of Compare Load Registers

The CMPLD1, CMPLD2 and CCCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. A compare register can be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there is the possibility that the counter may have already counted past the new compare value by the time the compare register is updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are updated in hardware in the same way the counter register is re-initialized to the value stored in the LOAD register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See Figure 39-8.

39.4.2.15 MODULO COUNTING Mode

To create a modulo counter using COMP1 and COMP2 as the counter boundaries (instead of $0000 and $FFFF), set the registers in the following manner. Set CNTMODE to either 100 (quadrature count mode) or 101 (count with direction mode). Use count through roll-over (LENGTH = 0) and continuous count (ONCE = 0). Set COMP1 and CMPLD1 to the upper boundary value. Set COMP2 and CMPLD2 to the lower boundary value. Set CMPMODE = 10 (COMP1 is used when counting up and COMP2 is used when counting down). Set CLC2 = 110 (load CNTR with value of CMPLD2 on COMP1 compare) and CLC1 = 111 (load CNTR with value of CMPLD1 on COMP2 compare).

39.4.2.16 Compare Register and OFLAG Operation

The compare flags and output flag are registered signals and can only change state after the counter reached the compare value. This means that, if the counter is continuously counting, then the compare flag is set on the count after the counter reaches the compare value. When counting is not continuous (such as when using a prescaler or when counting edges on the primary input), the compare flag or OFLAG are not updated until the counter is about to transition to the next count. This means that if the compare value is 4, then the flags will be changed on the same clock edge that transitions the counter from 4 to its next value.

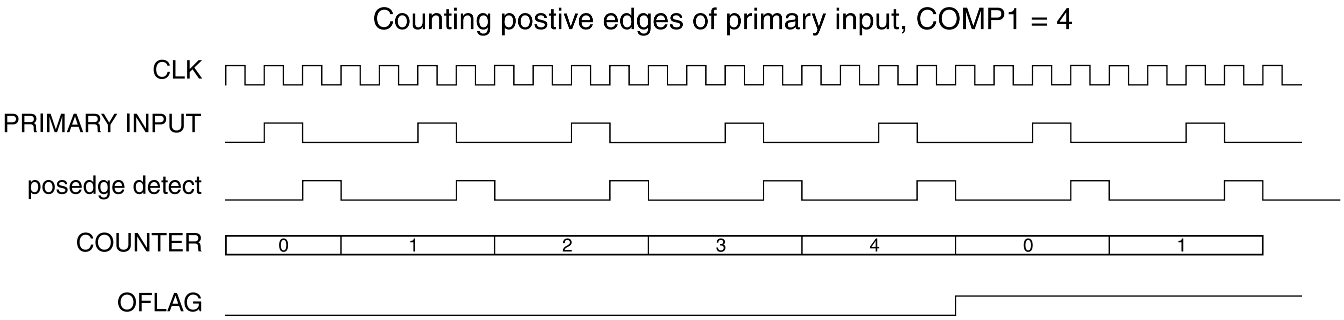


Figure 39-10. OFLAG timing during non-continuous counting

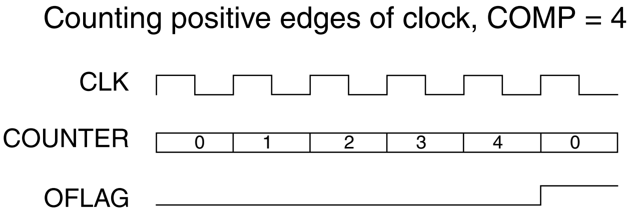


Figure 39-11. OFLAG timing during continuous counting

39.4.3 Other Features

This section describes other features of eTimer.

39.4.3.1 Redundant OFLAG Checking

This mode allows the user to bundle two timer functions generating any pattern to compare their resulting OFLAG behaviors (output signal).

The redundant mode is used to support online checks for reliability and functional safety reasons. Whenever a mismatch between the two adjacent channels occurs, it is reported via an interrupt to the core and the two outputs are put into their inactive states. An error is flagged via the RCF flag.

This feature can be tested by forcing a transition on one of the OFLAGs using the VAL and FORCE bits of the channel.

39.4.3.2 Loopback Checking

This mode is always available in that one channel can generate an OFLAG while another channel uses the first channels OFLAG as its input to be measured and verified to be as expected.

39.4.3.3 Input Capture Mode

Input capture is used to measure pulse width (by capturing the counter value on two successive input edges) or waveform period (by capturing the counter value on two consecutive rising edges or two consecutive falling edges). The Capture Registers store a copy of the counter's value when an input edge (positive, negative, or both) is detected. The type of edge to be captured by each circuit is determined by the CPT1MODE and CPT2MODE bits whose functionality is listed in CPT1MODE. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

39.4.3.4 Master/Slave Mode

Any timer channel can be assigned as a Master (MSTR = 1). A Master's compare signal can be broadcasted to the other channels within the module. The other counters can be configured to reinitialize their counters (COINIT = 1) and/or force their OFLAG output signals (COFRC = 1) to predetermined values when a Master counter compare event occurs.

39.4.3.5 Watchdog Timer

NOTE

> The watchdog timer may not be implemented on each eTimer module instance.

The watchdog timer is used to monitor for a stalled count when channel 0 is in quadrature count mode. When the watchdog is enabled, it loads the time-out value into a down counter. The down counter counts as long as channel 0 is in quadrature decode count mode. If this down counter reaches zero, an interrupt is asserted. The down counter is reloaded to the time-out value each time the counter value from channel 0 changes. If the channel 0 count value is toggling between two values (indicating a possibly stalled encoder), then the down counter is not reloaded.

39.5 Resets

The eTimer module can only be reset by a full system reset. This forces all registers to their reset state and clears the OFLAG signals if they are asserted. The counters will be turned off until the settings in the CTRL registers are changed.

39.6 Clocks

Each timer channel receives its own copy of the peripheral bus clock. This allows for better power management by turning off the clock to unused channels. The watchdog timer (if implemented) has its own version of this clock. Each bit of the ENBL register uses the clock for that corresponding channel. The DREQn registers use the clock for channel, 0 but this clock can be turned off after these registers have been programmed.

39.7 Interrupts

The following can generate interrupts within the eTimer.

- Watchdog timer (if implemented)
- Each channel can generate an interrupt from several sources

The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 39-1. Interrupt summary

Core interrupt	Interrupt flag	Interrupt enable	Name	Description
Channels 0-n	TCF	TCFIE	Compare interrupt	Compare of counter and related compare register
	TCF1	TCF1IE	Compare 1 interrupt	Compare of the counter and COMP1 register
	TCF2	TCF2IE	Compare 2 interrupt	Compare of the counter and COMP2 register
	TOF	TOFIE	Overflow interrupt	Generated on counter roll-over or roll-under
	IELF	IELFIE	Input Low Edge interrupt	Falling edge of the secondary input signal
	IEHF	IEHFIE	Input High Edge interrupt	Rising edge of the secondary input signal
	ICF1	ICF1IE	Input Capture 1 interrupt	Input capture event for CAPT1
	ICF2	ICF2IE	Input Capture 2 interrupt	Input capture event for CAPT2
Watchdog[1]	WDF	WDFIE	Watchdog time-out interrupt	Watchdog has timed out
Redundant Channel Checking	RCF	RCFIE	Redundant Channel Fault interrupt	Miscompare with redundant channel

[1] If applicable. Not all instantiations of eTimer implement a watchdog timer.

39.8 DMA

Each channel can request a DMA read access for each of the capture registers and a DMA write request for each of the compare preload registers. The DREQ registers select amongst these 32 DMA request sources to generate the 4 top level DMA request outputs.

Table 39-2. DMA Summary

DMA Request	DMA Enable	Name	Description
Channels 0-n	ICF1DE	CAPT1 read request	CAPT1 contains a value
	ICF2DE	CAPT2 read request	CAPT2 contains a value
	CMPLD1DE	CMPLD1 write request	CMPLD1 needs an update
	CMPLD2DE	CMPLD2 write request	CMPLD2 needs an update

Chapter 40
Motor Control Pulse Width Modulator Module (FlexPWM)

40.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

This section contains features, modes of operation, and block diagrams of the FlexPWM.

40.1.1 Overview

The pulse width modulator module (FlexPWM) contains one or more PWM submodules, each capable of controlling a single half-bridge power stage. One or more fault channels may also be included.

This PWM is capable of controlling most motor types: AC induction motors (ACIM), Permanent Magnet AC motors (PMAC), both brushless (BLDC) and brush DC motors (BDC), switched (SRM) and variable reluctance motors (VRM), and stepper motors.

40.1.2 Features

The FlexPWM module provides the following features:

- 16 bits of resolution for center, edge aligned, and asymmetrical PWMs
- PWM outputs capable of operating as complementary pairs or independent channels
- Can accept signed numbers for PWM generation
- Independent control of both edges of each PWM output

- Synchronization to external hardware or other PWM supported
- Double buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software control for each PWM output
- All outputs can be programmed to change simultaneously via a "Force Out" event
- PWMX pin can optionally output a third PWM signal from each submodule
- PWMX channels not used for PWM generation can be used for buffered output compare functions
- Channels not used for PWM generation can be used for input capture functions
- Capture is supported only for X channels not A and B channels
- Enhanced dual edge capture functionality
- The option to supply the source for each complementary PWM signal pair from any of the following:
  - External digital pin
  - Internal timer channel
  - External ADC input, taking into account values set in ADC high and low limit registers

40.1.3 Modes of operation

Care must be exercised when using this module in certain chip operating modes. Some motors (such 3-phase AC motors) require regular software updates for proper operation. Failure to do so could result in destroying the motor or inverter. Because of this, PWM outputs are placed in their inactive states in STOP mode, and optionally under debug mode. PWM outputs are reactivated (assuming they were active to begin with) when these modes are exited.

Table 40-1. Modes when PWM operation is restricted

Mode	Description
STOP	Peripheral and CPU clocks are stopped. PWM outputs are driven inactive.
DEBUG	CPU and peripheral clocks continue to run, but CPU maybe stalled for periods of time. PWM outputs are driven inactive as a function of the DBGEN bit.

40.1.4 Block Diagrams

This section contains block diagrams of the FlexPWM.

40.1.4.1 Module level

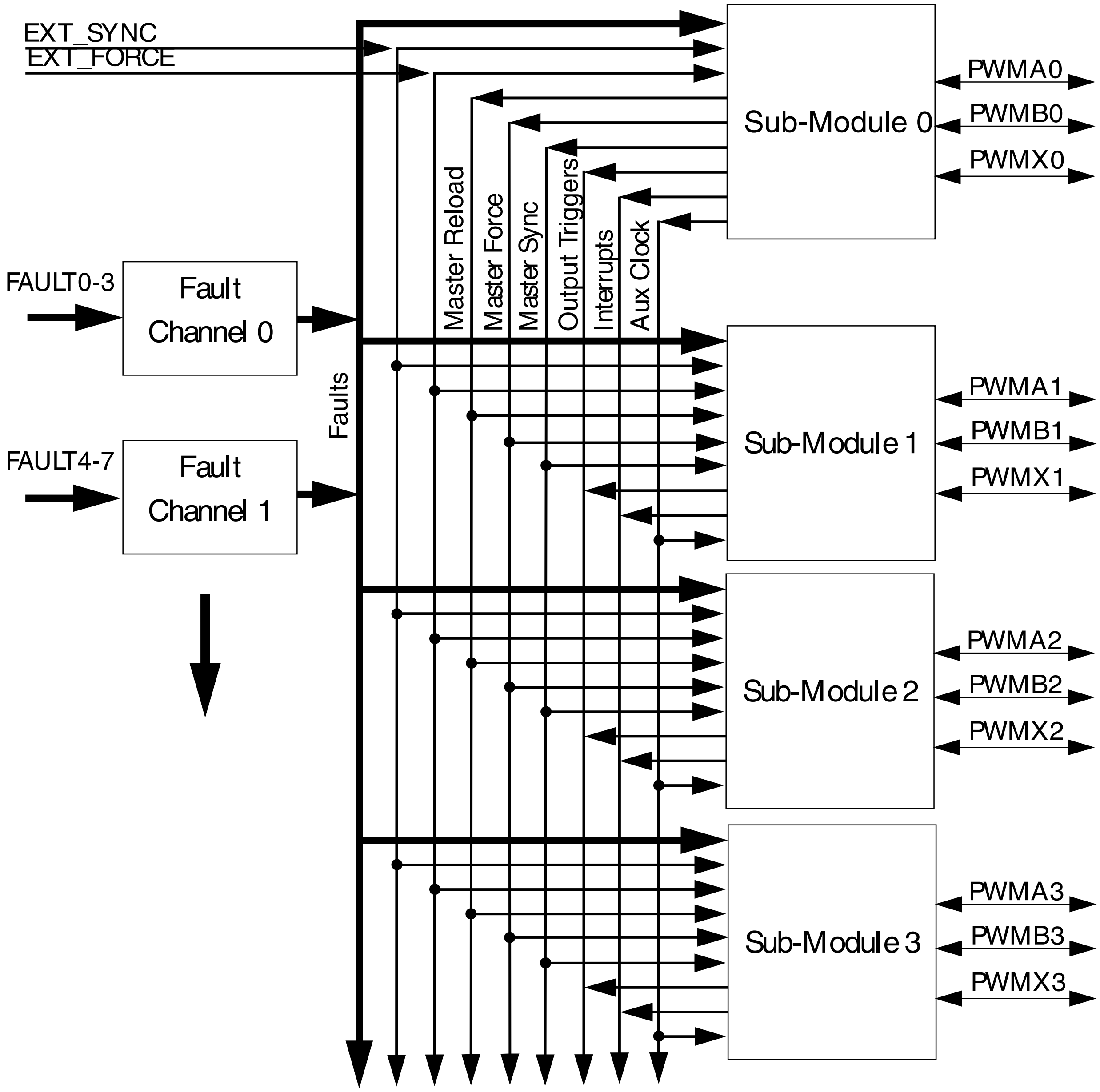


Figure 40-1. PWM block diagram

40.1.4.2 PWM submodule

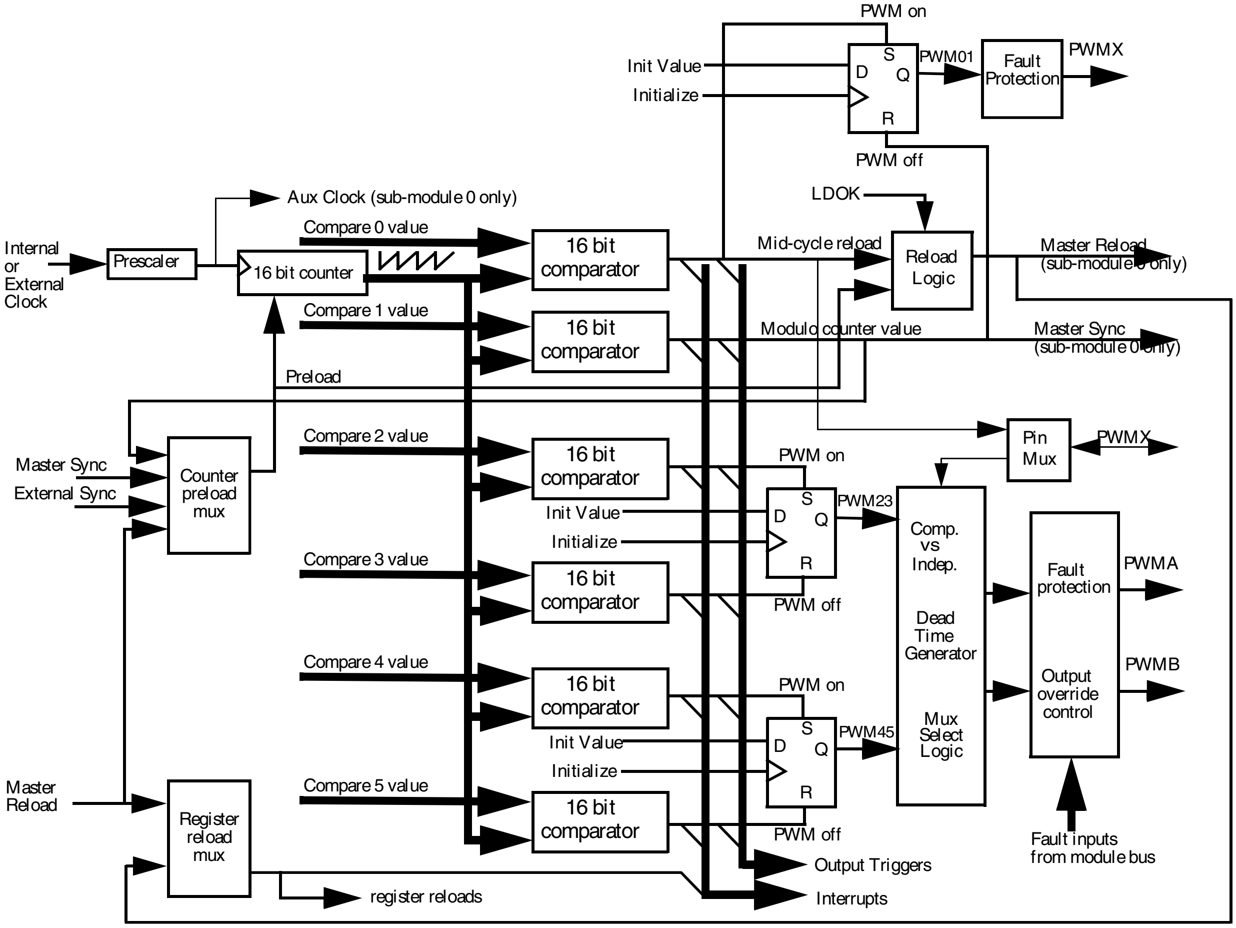


Figure 40-2. PWM submodule block diagram

40.2 External Signal Descriptions

The PWM module has external pins named PWMA[n], PWMB[n], PWMX[n], FAULT[n], EXT_SYNC, EXT_FORCE, EXTA[n], and EXTB[n]. The PWM module also has an on chip input called EXT_CLK and an on-chip output called OUT_TRIG[n].

40.2.1 PWMA[n] and PWMB[n] - External PWM Pair

These pins are the output pins of the PWM channels. They can be independent PWM signals or a complementary pair.

40.2.2 PWMX[n] - Auxiliary PWM signal

These pins are the auxiliary output pins of the PWM channels. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or they can be used to generate the IPOL bit during deadtime correction.

40.2.3 FAULT[n] - Fault inputs

These are input pins for disabling selected PWM outputs.

40.2.4 EXT_SYNC - External Synchronization Signal

This input signal allows a source external to the PWM to initialize the PWM counter. In this manner the PWM can be synchronized to external circuitry.

40.2.5 EXT_FORCE - External Output Force Signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. In this manner the PWM can be synchronized to external circuitry. An example would be to simultaneously switch all of the PWM outputs on a commutation boundary for trapezoidal control of a BLDC motor. The boundary can be established via external logic or an on-chip timer.

40.2.6 EXTA[n] and EXTB[n] - Alternate PWM Control Signals

These pins allow an alternate source to control the PWMA and PWMB outputs although typically, the EXTA input will be used for the generation of a complementary pair. Typical connections include ADC results registers, TMR outputs, GPIO inputs, and comparator outputs.

40.2.7 OUT_TRIG0[n] and OUT_TRIG1[n] - Output Triggers

These outputs allow the PWM submodules to control timing of the ADC conversions. See Submodule n Output Trigger Control Register (FlexPWM_SUB*n*_TCTRL) for a description of how to enable these outputs and how the compare registers match up to the output triggers.

40.2.8 EXT_CLK - External Clock Signal

This on-chip input signal allows an on-chip source external to the PWM (typically a Timer) to control the PWM clocking. In this manner the PWM can be synchronized to the Timer. This signal must be generated synchronously to the PWM's clock since it is not resynchronized in the PWM.

40.2.9 EXT_SWITCH - External Switch Signal

This input signal allows a source external to the PWM to control the PWM generation source within submodule1. This signal only functions on submodule1 and selects the source of the generated PWM signal to be either submodule1 or submodule0.

40.3 Memory Map and Registers

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level and the address offset is defined at the module level. There are a set of registers for each PWM submodule, for the configuration logic, and for each Fault channel.

Registers are only accessible using 16- or 32-bit accesses. Byte accesses are not allowed.

Submodule registers are repeated for each PWM submodule. The base address of submodule 0 is the same as the base address for the PWM module as a whole. The base address of submodule 1 is 50h. This is the base address of the PWM module plus an offset based on the number of registers in a submodule. The base address of submodule 2 is equal to the base address of submodule 1 plus this same offset.

The base address of the configuration registers is equal to the base address of the FlexPWM plus as offset of 140h.

The base address of the fault logic is equal to the base address of the FlexPWM plus an offset of 14Ch.

NOTE

A write access to the read-only Capture Value and Capture Value Cycle registers will result in a bus error.

FlexPWM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Submodule n Counter Register (FlexPWM_SUB0_CNT)	16	R	0000h	40.3.1/1132
2	Submodule n Initial Count Register (FlexPWM_SUB0_INIT)	16	R/W	0000h	40.3.2/1132
4	Submodule n Control 2 Register (FlexPWM_SUB0_CTRL2)	16	R/W	0000h	40.3.3/1133
6	Submodule n Control 1 Register (FlexPWM_SUB0_CTRL1)	16	R/W	0400h	40.3.4/1135
8	Submodule n Value Register 0 (FlexPWM_SUB0_VAL0)	16	R/W	0000h	40.3.5/1137
A	Submodule n Value Register 1 (FlexPWM_SUB0_VAL1)	16	R/W	0000h	40.3.6/1138
C	Submodule n Value Register 2 (FlexPWM_SUB0_VAL2)	16	R/W	0000h	40.3.7/1139
E	Submodule n Value Register 3 (FlexPWM_SUB0_VAL3)	16	R/W	0000h	40.3.8/1139
10	Submodule n Value Register 4 (FlexPWM_SUB0_VAL4)	16	R/W	0000h	40.3.9/1140
12	Submodule n Value Register 5 (FlexPWM_SUB0_VAL5)	16	R/W	0000h	40.3.10/ 1141
18	Submodule n Output Control Register (FlexPWM_SUB0_OCTRL)	16	R/W	See section	40.3.11/ 1141
1A	Submodule n Status Register (FlexPWM_SUB0_STS)	16	R/W	0000h	40.3.12/ 1143
1C	Submodule n Interrupt Enable Register (FlexPWM_SUB0_INTEN)	16	R/W	0000h	40.3.13/ 1144
1E	Submodule n DMA Enable Register (FlexPWM_SUB0_DMAEN)	16	R/W	0000h	40.3.14/ 1146
20	Submodule n Output Trigger Control Register (FlexPWM_SUB0_TCTRL)	16	R/W	0000h	40.3.15/ 1147
22	Submodule n Fault Disable Mapping Register (FlexPWM_SUB0_DISMAP)	16	R/W	FFFFh	40.3.16/ 1148
24	Submodule n Deadtime Count Register 0 (FlexPWM_SUB0_DTCNT0)	16	R/W	07FFh	40.3.17/ 1149
26	Submodule n Deadtime Count Register 1 (FlexPWM_SUB0_DTCNT1)	16	R/W	07FFh	40.3.18/ 1150
30	Submodule n Capture Control X Register (FlexPWM_SUB0_CAPTCTRLX)	16	R/W	0000h	40.3.19/ 1151
32	Submodule n Capture Compare X Register (FlexPWM_SUB0_CAPTCMPX)	16	R/W	0000h	40.3.20/ 1152
34	Submodule n Capture Value 0 Register (FlexPWM_SUB0_CVAL0)	16	R	0000h	40.3.21/ 1153
36	Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB0_CVAL0CYC)	16	R	0000h	40.3.22/ 1154
38	Submodule n Capture Value 1 Register (FlexPWM_SUB0_CVAL1)	16	R	0000h	40.3.23/ 1154
3A	Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB0_CVAL1CYC)	16	R	0000h	40.3.24/ 1155
50	Submodule n Counter Register (FlexPWM_SUB1_CNT)	16	R	0000h	40.3.1/1132
52	Submodule n Initial Count Register (FlexPWM_SUB1_INIT)	16	R/W	0000h	40.3.2/1132
54	Submodule n Control 2 Register (FlexPWM_SUB1_CTRL2)	16	R/W	0000h	40.3.3/1133
56	Submodule n Control 1 Register (FlexPWM_SUB1_CTRL1)	16	R/W	0400h	40.3.4/1135

Table continues on the next page...

FlexPWM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
58	Submodule n Value Register 0 (FlexPWM_SUB1_VAL0)	16	R/W	0000h	40.3.5/1137
5A	Submodule n Value Register 1 (FlexPWM_SUB1_VAL1)	16	R/W	0000h	40.3.6/1138
5C	Submodule n Value Register 2 (FlexPWM_SUB1_VAL2)	16	R/W	0000h	40.3.7/1139
5E	Submodule n Value Register 3 (FlexPWM_SUB1_VAL3)	16	R/W	0000h	40.3.8/1139
60	Submodule n Value Register 4 (FlexPWM_SUB1_VAL4)	16	R/W	0000h	40.3.9/1140
62	Submodule n Value Register 5 (FlexPWM_SUB1_VAL5)	16	R/W	0000h	40.3.10/ 1141
68	Submodule n Output Control Register (FlexPWM_SUB1_OCTRL)	16	R/W	See section	40.3.11/ 1141
6A	Submodule n Status Register (FlexPWM_SUB1_STS)	16	R/W	0000h	40.3.12/ 1143
6C	Submodule n Interrupt Enable Register (FlexPWM_SUB1_INTEN)	16	R/W	0000h	40.3.13/ 1144
6E	Submodule n DMA Enable Register (FlexPWM_SUB1_DMAEN)	16	R/W	0000h	40.3.14/ 1146
70	Submodule n Output Trigger Control Register (FlexPWM_SUB1_TCTRL)	16	R/W	0000h	40.3.15/ 1147
72	Submodule n Fault Disable Mapping Register (FlexPWM_SUB1_DISMAP)	16	R/W	FFFFh	40.3.16/ 1148
74	Submodule n Deadtime Count Register 0 (FlexPWM_SUB1_DTCNT0)	16	R/W	07FFh	40.3.17/ 1149
76	Submodule n Deadtime Count Register 1 (FlexPWM_SUB1_DTCNT1)	16	R/W	07FFh	40.3.18/ 1150
80	Submodule n Capture Control X Register (FlexPWM_SUB1_CAPTCTRLX)	16	R/W	0000h	40.3.19/ 1151
82	Submodule n Capture Compare X Register (FlexPWM_SUB1_CAPTCMPX)	16	R/W	0000h	40.3.20/ 1152
84	Submodule n Capture Value 0 Register (FlexPWM_SUB1_CVAL0)	16	R	0000h	40.3.21/ 1153
86	Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB1_CVAL0CYC)	16	R	0000h	40.3.22/ 1154
88	Submodule n Capture Value 1 Register (FlexPWM_SUB1_CVAL1)	16	R	0000h	40.3.23/ 1154
8A	Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB1_CVAL1CYC)	16	R	0000h	40.3.24/ 1155
A0	Submodule n Counter Register (FlexPWM_SUB2_CNT)	16	R	0000h	40.3.1/1132
A2	Submodule n Initial Count Register (FlexPWM_SUB2_INIT)	16	R/W	0000h	40.3.2/1132
A4	Submodule n Control 2 Register (FlexPWM_SUB2_CTRL2)	16	R/W	0000h	40.3.3/1133
A6	Submodule n Control 1 Register (FlexPWM_SUB2_CTRL1)	16	R/W	0400h	40.3.4/1135
A8	Submodule n Value Register 0 (FlexPWM_SUB2_VAL0)	16	R/W	0000h	40.3.5/1137
AA	Submodule n Value Register 1 (FlexPWM_SUB2_VAL1)	16	R/W	0000h	40.3.6/1138
AC	Submodule n Value Register 2 (FlexPWM_SUB2_VAL2)	16	R/W	0000h	40.3.7/1139
AE	Submodule n Value Register 3 (FlexPWM_SUB2_VAL3)	16	R/W	0000h	40.3.8/1139

Table continues on the next page...

FlexPWM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B0	Submodule n Value Register 4 (FlexPWM_SUB2_VAL4)	16	R/W	0000h	40.3.9/1140
B2	Submodule n Value Register 5 (FlexPWM_SUB2_VAL5)	16	R/W	0000h	40.3.10/ 1141
B8	Submodule n Output Control Register (FlexPWM_SUB2_OCTRL)	16	R/W	See section	40.3.11/ 1141
BA	Submodule n Status Register (FlexPWM_SUB2_STS)	16	R/W	0000h	40.3.12/ 1143
BC	Submodule n Interrupt Enable Register (FlexPWM_SUB2_INTEN)	16	R/W	0000h	40.3.13/ 1144
BE	Submodule n DMA Enable Register (FlexPWM_SUB2_DMAEN)	16	R/W	0000h	40.3.14/ 1146
C0	Submodule n Output Trigger Control Register (FlexPWM_SUB2_TCTRL)	16	R/W	0000h	40.3.15/ 1147
C2	Submodule n Fault Disable Mapping Register (FlexPWM_SUB2_DISMAP)	16	R/W	FFFFh	40.3.16/ 1148
C4	Submodule n Deadtime Count Register 0 (FlexPWM_SUB2_DTCNT0)	16	R/W	07FFh	40.3.17/ 1149
C6	Submodule n Deadtime Count Register 1 (FlexPWM_SUB2_DTCNT1)	16	R/W	07FFh	40.3.18/ 1150
D0	Submodule n Capture Control X Register (FlexPWM_SUB2_CAPTCTRLX)	16	R/W	0000h	40.3.19/ 1151
D2	Submodule n Capture Compare X Register (FlexPWM_SUB2_CAPTCMPX)	16	R/W	0000h	40.3.20/ 1152
D4	Submodule n Capture Value 0 Register (FlexPWM_SUB2_CVAL0)	16	R	0000h	40.3.21/ 1153
D6	Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB2_CVAL0CYC)	16	R	0000h	40.3.22/ 1154
D8	Submodule n Capture Value 1 Register (FlexPWM_SUB2_CVAL1)	16	R	0000h	40.3.23/ 1154
DA	Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB2_CVAL1CYC)	16	R	0000h	40.3.24/ 1155
F0	Submodule n Counter Register (FlexPWM_SUB3_CNT)	16	R	0000h	40.3.1/1132
F2	Submodule n Initial Count Register (FlexPWM_SUB3_INIT)	16	R/W	0000h	40.3.2/1132
F4	Submodule n Control 2 Register (FlexPWM_SUB3_CTRL2)	16	R/W	0000h	40.3.3/1133
F6	Submodule n Control 1 Register (FlexPWM_SUB3_CTRL1)	16	R/W	0400h	40.3.4/1135
F8	Submodule n Value Register 0 (FlexPWM_SUB3_VAL0)	16	R/W	0000h	40.3.5/1137
FA	Submodule n Value Register 1 (FlexPWM_SUB3_VAL1)	16	R/W	0000h	40.3.6/1138
FC	Submodule n Value Register 2 (FlexPWM_SUB3_VAL2)	16	R/W	0000h	40.3.7/1139
FE	Submodule n Value Register 3 (FlexPWM_SUB3_VAL3)	16	R/W	0000h	40.3.8/1139
100	Submodule n Value Register 4 (FlexPWM_SUB3_VAL4)	16	R/W	0000h	40.3.9/1140
102	Submodule n Value Register 5 (FlexPWM_SUB3_VAL5)	16	R/W	0000h	40.3.10/ 1141
108	Submodule n Output Control Register (FlexPWM_SUB3_OCTRL)	16	R/W	See section	40.3.11/ 1141

Table continues on the next page...

FlexPWM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10A	Submodule n Status Register (FlexPWM_SUB3_STS)	16	R/W	0000h	40.3.12/ 1143
10C	Submodule n Interrupt Enable Register (FlexPWM_SUB3_INTEN)	16	R/W	0000h	40.3.13/ 1144
10E	Submodule n DMA Enable Register (FlexPWM_SUB3_DMAEN)	16	R/W	0000h	40.3.14/ 1146
110	Submodule n Output Trigger Control Register (FlexPWM_SUB3_TCTRL)	16	R/W	0000h	40.3.15/ 1147
112	Submodule n Fault Disable Mapping Register (FlexPWM_SUB3_DISMAP)	16	R/W	FFFFh	40.3.16/ 1148
114	Submodule n Deadtime Count Register 0 (FlexPWM_SUB3_DTCNT0)	16	R/W	07FFh	40.3.17/ 1149
116	Submodule n Deadtime Count Register 1 (FlexPWM_SUB3_DTCNT1)	16	R/W	07FFh	40.3.18/ 1150
120	Submodule n Capture Control X Register (FlexPWM_SUB3_CAPTCTRLX)	16	R/W	0000h	40.3.19/ 1151
122	Submodule n Capture Compare X Register (FlexPWM_SUB3_CAPTCMPX)	16	R/W	0000h	40.3.20/ 1152
124	Submodule n Capture Value 0 Register (FlexPWM_SUB3_CVAL0)	16	R	0000h	40.3.21/ 1153
126	Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB3_CVAL0CYC)	16	R	0000h	40.3.22/ 1154
128	Submodule n Capture Value 1 Register (FlexPWM_SUB3_CVAL1)	16	R	0000h	40.3.23/ 1154
12A	Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB3_CVAL1CYC)	16	R	0000h	40.3.24/ 1155
140	Output Enable Register (FlexPWM_OUTEN)	16	R/W	0000h	40.3.25/ 1155
142	Mask Register (FlexPWM_MASK)	16	R/W	0000h	40.3.26/ 1157
144	Software Controlled Output Register (FlexPWM_SWCOUT)	16	R/W	0000h	40.3.27/ 1158
146	Deadtime Source Select Register (FlexPWM_DTSRCSEL)	16	R/W	0000h	40.3.28/ 1160
148	Master Control Register (FlexPWM_MCTRL)	16	R/W	0000h	40.3.29/ 1162
14A	Reserved Register (FlexPWM_RESERVED)	16	R (reads 0)	0000h	40.3.30/ 1164
14C	Fault Control Register (FlexPWM_FCTRL)	16	R/W	0000h	40.3.30/ 1164
14E	Fault Status Register (FlexPWM_FSTS)	16	R/W	See section	40.3.31/ 1166
150	Fault Filter Register (FlexPWM_FFILT)	16	R/W	0000h	40.3.32/ 1167

40.3.1 Submodule n Counter Register (FlexPWM_SUB*n*_CNT)

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 0h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CNT															
Write	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CNT field descriptions

Field	Description
0–15 CNT	Count State of the signed 16-bit submodule counter

40.3.2 Submodule n Initial Count Register (FlexPWM_SUB*n*_INIT)

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of INIT_SEL) or when FORCE is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

NOTE

> The INIT register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. This register cannot be written when LDOK is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 2h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	INIT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_INIT field descriptions

Field	Description
0–15 INIT	Initial count Initial count value for the PWM in PWM clock periods

40.3.3 Submodule n Control 2 Register (FlexPWM_SUB*n*_CTRL2)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 4h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read Write	DBGEN	Reserved	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL	
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
Write		FORCE						
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CTRL2 field descriptions

Field	Description
0 DBGEN	Debug Enable When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in debug mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in debug mode. The key point is PWM parameter updates will not occur in debug mode. Any motors requiring such updates should be disabled during debug mode. If in doubt, leave this bit set to zero.
1 Reserved	This field is reserved.
2 INDEP	Independent or Complementary Pair Operation

Table continues on the next page...

FlexPWM_SUBn_CTRL2 field descriptions (continued)

Field	Description
	This bit determines if the PWMA and PWMB channels will be independent PWMs or a complementary PWM pair. 0 PWMA and PWMB form a complementary PWM pair. 1 PWMA and PWMB outputs are independent PWMs.
3 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT, (see PWM Generation), is asserted.
4 PWM45_INIT	PWM45 Initial Value This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT, (see PWM Generation), is asserted.
5 PWMX_INIT	PWMX Initial Value This read/write bit determines the initial value for PWMX and the value to which it is forced when FORCE_INIT, (see PWM Generation), is asserted.
6–7 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00 Local sync (PWMX) causes initialization. 01 Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 10 Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 11 EXT_SYNC causes initialization.
8 FRCEN	Force Initialization Enable This bit allows the FORCE_OUT signal to initialize the counter without regard to the signal selected by INIT_SEL. This is a software controlled initialization. 0 Initialization from a Force Out event is disabled. 1 Initialization from a Force Out event is enabled.
9 FORCE	Force Initialization If the FORCE_SEL bits are set to 000, writing a 1 to this bit results in a Force Out event. This causes the following actions to be taken: • The PWMA and PWMB output pins will assume values based on the SEL23 and SEL45 bits. • If the FRCEN bit is set, the counter value will be initialized with the INIT register value.
10–12 FORCE_SEL	Force Source Select This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000 The local force signal, FORCE, from this submodule is used to force updates. 001 The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0. 010 The local reload signal from this submodule is used to force updates without regard to the state of LDOK. 011 The master reload signal from submodule0 is used to force updates if LDOK is set. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 100 The local sync signal from this submodule is used to force updates.

Table continues on the next page...

FlexPWM_SUB*n*_CTRL2 field descriptions (continued)

Field	Description
	101 The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 110 The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111 reserved
13 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, the LDOK bit in submodule 0 should be used since the local LDOK bit will be ignored. 0 The local RELOAD signal is used to reload registers. 1 The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0.
14–15 CLK_SEL	Clock Source Select These read/write bits determine the source of the clock signal for this submodule. 00 The peripheral clock is used as the clock for the local prescaler and counter. 01 EXT_CLK is used as the clock for the local prescaler and counter. 10 Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11 reserved

40.3.4 Submodule n Control 1 Register (FlexPWM_SUB*n*_CTRL1)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 6h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	LDFQ				HALF	FULL	DT	
Write								
Reset	0	0	0	0	0	1	0	0

Bit	8	9	10	11	12	13	14	15
Read	0	PRSC			PHSSHFT	LDMOD	0	DBLEN
Write								
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CTRL1 field descriptions

Field	Description
0–3 LDFQ	Load Frequency

Table continues on the next page...

FlexPWM_SUB*n*_CTRL1 field descriptions (continued)

Field	Description
	These buffered read/write bits select the PWM load frequency. The PWM reload frequency is at every LDFQ+1 PWM opportunities. Reset clears the LDFQ bits, selecting loading every PWM opportunity. A PWM opportunity is determined by the HALF and FULL bits. **NOTE:** The LDFQx bits take effect when the current load cycle is complete regardless of the state of the LDOK bit. Reading the LDFQx bits reads the buffered values and not necessarily the values currently in effect.
4 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the SUB*n*_VAL0 register and does not have to be half way through the PWM cycle. 0 Half-cycle reloads disabled. 1 Half-cycle reloads enabled.
5 FULL	Full Cycle Reload This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the SUB*n*_VAL1 register. Either the HALF or FULL bit must be set in order to move the buffered data into the registers used by the PWM generators. If both the HALF and FULL bits are set, then reloads can occur twice per cycle. 0 Full-cycle reloads disabled. 1 Full-cycle reloads enabled.
6–7 DT	Deadtime These read only bits reflect the sampled values of the PWMX input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CLK_SEL. **NOTE:** Reading the PRSCx bits reads the buffered values and not necessarily the values currently in effect. The PRSCx bits take effect at the beginning of the next PWM cycle and only when the load okay bit, LDOK, is set or LDMOD is set. This field cannot be written when LDOK is set. 000 PWM clock frequency = f_{CLK} 001 PWM clock frequency = f_{CLK} / 2 010 PWM clock frequency = f_{CLK} / 4 011 PWM clock frequency = f_{CLK} / 8 100 PWM clock frequency = f_{CLK} / 16 101 PWM clock frequency = f_{CLK} / 32 110 PWM clock frequency = f_{CLK} / 64 111 PWM clock frequency = f_{CLK} / 128
12 PHSSHFT	Phase Shift Mode Enable This read/write bit is only functional in submodule 0. It is used to enable the EXT_SWITCH input to select the source of the generated PWM23/PWM45 signals within submodule 0. If this enable is clear or the EXT_SWITCH signal is 0, then the PWM23/PWM45 signals generated within submodule 0 are used by the force out logic in submodule 0. If this enable is set and the EXT_SWITCH signal is also set, then the PWM23/PWM45 signals from submodule 1 are used by the force out logic in submodule 0.

Table continues on the next page...

FlexPWM_SUB*n*_CTRL1 field descriptions (continued)

Field	Description
	0 Phase shifted mode is disabled. The EXT_SWITCH input is ignored. Normal operation. 1 Phase shifted mode is enabled. The EXT_SWITCH input is used to select generated PWM source for force out logic in submodule 0.
13 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule. 0 Buffered registers of this submodule are loaded and take effect at the next PWM reload if LDOK is set. 1 Buffered registers of this submodule are loaded and take effect immediately upon LDOK being set.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior. 0 Double switching disabled. 1 Double switching enabled.

40.3.5 Submodule n Value Register 0 (FlexPWM_SUB*n*_VAL0)

The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWMX signal is set and the local sync signal is reset. This register is not byte accessible.

NOTE

> The SUB*n*_VAL0 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL0 cannot be written when LDOK is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 8h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL0 field descriptions

Field	Description
0–15 VAL0	Value 0 Mid-cycle reload point for the PWM in PWM clock periods

40.3.6 Submodule n Value Register 1 (FlexPWM_SUB*n*_VAL1)

The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter will reload itself with the contents of the INIT register and assert the local sync signal while resetting PWMX. This register is not byte accessible.

NOTE

The SUB*n*_VAL1 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL1 cannot be written when LDOK is set. Reading VAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

NOTE

If SUB*n*_VAL1 register defines the timer period (Local Sync is selected as the counter initialization signal), a 100% duty cycle cannot be achieved on the PWMX output. The PWMX output will be low for a minimum of one count every cycle after the count reaches VAL1. When the Master Sync signal (only originated by the Local Sync from sub-module 0) is used to control the timer period, the SUB*n*_VAL1 register can be free for other functions such as PWM generation without duty cycle limitation.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + Ah offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL1 field descriptions

Field	Description
0–15 VAL1	Value 1 Modulo count value (maximum count) for the submodule counter

40.3.7 Submodule n Value Register 2 (FlexPWM_SUB*n*_VAL2)

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high (PWM submodule). This register is not byte accessible.

NOTE

> The SUB*n*_VAL2 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL2 cannot be written when LDOK is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + Ch offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL2 field descriptions

Field	Description
0–15 VAL2	Value 2 Count value to set PWM23 high

40.3.8 Submodule n Value Register 3 (FlexPWM_SUB*n*_VAL3)

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low (PWM submodule). This register is not byte accessible.

NOTE

> The SUB*n*_VAL3 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL3 cannot be written when

LDOK is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + Eh offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL3 field descriptions

Field	Description
0–15 VAL3	Value 3 Count value to set PWM23 low

40.3.9 Submodule n Value Register 4 (FlexPWM_SUB*n*_VAL4)

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high (PWM submodule). This register is not byte accessible.

NOTE

The SUB*n*_VAL4 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL4 cannot be written when LDOK is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 10h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL4															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL4 field descriptions

Field	Description
0–15 VAL4	Count value to set PWM45 high

40.3.10 Submodule n Value Register 5 (FlexPWM_SUB*n*_VAL5)

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low (PWM submodule). This register is not byte accessible.

NOTE

The SUB*n*_VAL5 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL5 cannot be written when LDOK is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 12h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	VAL5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_VAL5 field descriptions

Field	Description
0–15 VAL5	Count value to set PWM45 low

40.3.11 Submodule n Output Control Register (FlexPWM_SUB*n*_OCTRL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 18h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	0	0	PWMX_IN	0		POLA	POLB	POLX
Write								
Reset	0	0	*	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		PWMAFS		PWMBFS		PWMXFS	
Write	▨	▨						
Reset	0	0	0	0	0	0	0	0

* Notes:

- PWMX_IN field: Undefined

FlexPWM_SUB*n*_OCTRL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 PWMX_IN	PWMX Input This read only bit shows the logic value currently being driven into the PWMX input.
3–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 POLA	PWMA Output Polarity This bit inverts the PWMA output polarity. 0 PWMA output not inverted. A high level on the PWMA pin represents the "on" or "active" state. 1 PWMA output inverted. A low level on the PWMA pin represents the "on" or "active" state.
6 POLB	PWMB Output Polarity This bit inverts the PWMB output polarity. 0 PWMB output not inverted. A high level on the PWMB pin represents the "on" or "active" state. 1 PWMB output inverted. A low level on the PWMB pin represents the "on" or "active" state.
7 POLX	PWMX Output Polarity This bit inverts the PWMX output polarity. 0 PWMX output not inverted. A high level on the PWMX pin represents the "on" or "active" state. 1 PWMX output inverted. A low level on the PWMX pin represents the "on" or "active" state.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–11 PWMAFS	PWMA Fault State These bits determine the fault state for the PWMA output during fault conditions and STOP mode . It may also define the output state during DEBUG modes depending on the settings of DBGEN. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
12–13 PWMBFS	PWMB Fault State These bits determine the fault state for the PWMB output during fault conditions and STOP mode . It may also define the output state during DEBUG modes depending on the settings of DBGEN.

Table continues on the next page...

FlexPWM_SUB*n*_OCTRL field descriptions (continued)

Field	Description
	00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
14–15 PWMXFS	PWMX Fault State These bits determine the fault state for the PWMX output during fault conditions and STOP mode . It may also define the output state during DEBUG modes depending on the settings of DBGEN. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.

40.3.12 Submodule n Status Register (FlexPWM_SUB*n*_STS)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 1Ah offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	RUF	REF	RF	0				CFX1	CFX0	CMPF					
Write			w1c	w1c					w1c	w1c	w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_STS field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 RUF	Registers Updated Flag This read-only flag is set when SUB*n*_INIT, SUB*n*_VAL*x*, or SUB*n*_CTRL*0x*[PRSC] has been written resulting in non-coherent data in the set of double-buffered registers. Clear RUF by a proper reload sequence consisting of a reload signal while LDOK = 1. Reset clears RUF. 0 No register update has occurred since last reload. 1 At least one of the double buffered registers has been updated since the last reload.
2 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while LDOK is 0 and the double buffered registers are in a non-coherent state (RUF = 1). Clear REF by writing a logic one to the REF bit. Reset clears REF. 0 No reload error occurred. 1 Reload signal occurred with non-coherent data and LDOK = 0.

Table continues on the next page...

FlexPWM_SUB*n*_STS field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>3
RF</td><td>Reload Flag
This read/write flag is set at the beginning of every reload cycle regardless of the state of the LDOK bit. Clear RF by writing a logic one to the RF bit when VALDE is clear (non-DMA mode). RF can also be cleared by the DMA done signal when VALDE is set (DMA mode). Reset clears RF.
0 No new reload cycle since last RF clearing
1 New reload cycle since last RF clearing</td></tr>
<tr><td>4–7
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>8
CFX1</td><td>Capture Flag X1
This bit is set when the word count of the Capture X1 FIFO (CX1CNT) exceeds the value of the CFXWM field. This bit is cleared by writing a one to this bit position if CX1DE is clear (non-DMA mode) or by the DMA done signal if CX1DE is set (DMA mode). Reset clears this bit.</td></tr>
<tr><td>9
CFX0</td><td>Capture Flag X0
This bit is set when the word count of the Capture X0 FIFO (CX0CNT) exceeds the value of the CFXWM field. This bit is cleared by writing a one to this bit position if CX0DE is clear (non-DMA mode) or by the DMA done signal if CX0DE is set (DMA mode). Reset clears this bit.</td></tr>
<tr><td>10–15
CMPF</td><td>Compare Flags
These bits are set when the submodule counter value matches the value of one of the SUB<i>n</i>_VAL<i>x</i> registers. Clear these bits by writing a 1 to a bit position.
<table>
<tr><th>CMPF bit</th><th>Value register</th></tr>
<tr><td>0</td><td>SUB<i>n</i>_VAL0</td></tr>
<tr><td>1</td><td>SUB<i>n</i>_VAL1</td></tr>
<tr><td>2</td><td>SUB<i>n</i>_VAL2</td></tr>
<tr><td>3</td><td>SUB<i>n</i>_VAL3</td></tr>
<tr><td>4</td><td>SUB<i>n</i>_VAL4</td></tr>
<tr><td>5</td><td>SUB<i>n</i>_VAL5</td></tr>
</table>
0 No compare event has occurred for a particular VAL<i>x</i> value.
1 A compare event has occurred for a particular VAL<i>x</i> value.</td></tr>
</table>

40.3.13 Submodule n Interrupt Enable Register (FlexPWM_SUB*n*_INTEN)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 1Ch offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	0		REIE	RIE	Reserved			
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CX1IE	CX0IE	CMPIE					
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_INTEN field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–1
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>2
REIE</td><td>Reload Error Interrupt Enable
This read/write bit enables the reload error flag (REF) to generate CPU interrupt requests. Reset clears RIE.
0 REF CPU interrupt requests disabled
1 REF CPU interrupt requests enabled</td></tr>
<tr><td>3
RIE</td><td>Reload Interrupt Enable
This read/write bit enables the reload flag (RF) to generate CPU interrupt requests. Reset clears RIE.
0 RF CPU interrupt requests disabled
1 RF CPU interrupt requests enabled</td></tr>
<tr><td>4–7
Reserved</td><td>This field is reserved.</td></tr>
<tr><td>8
CX1IE</td><td>Capture X 1 Interrupt Enable
This bit allows the CFX1 flag to create an interrupt request to the CPU. Do not set both this bit and the CX1DE bit.
0 Interrupt request disabled for CFX1.
1 Interrupt request enabled for CFX1.</td></tr>
<tr><td>9
CX0IE</td><td>Capture X 0 Interrupt Enable
This bit allows the CFX0 flag to create an interrupt request to the CPU. Do not set both this bit and the CX0DE bit.
0 Interrupt request disabled for CFX0.
1 Interrupt request enabled for CFX0.</td></tr>
<tr><td>10–15
CMPIE</td><td>Compare Interrupt Enables
These bits enable the CMPF flags to cause a compare interrupt request to the CPU.<table><tr><th>CMPIE bit</th><th>Value register</th></tr><tr><td>0</td><td>SUB<i>n</i>_VAL0</td></tr><tr><td>1</td><td>SUB<i>n</i>_VAL1</td></tr><tr><td>2</td><td>SUB<i>n</i>_VAL2</td></tr><tr><td>3</td><td>SUB<i>n</i>_VAL3</td></tr></table></td></tr>
</table>

Table continues on the next page...

FlexPWM_SUB*n*_INTEN field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>CMPIE bit</th><th>Value register</th></tr>
<tr><td>4</td><td>SUB<i>n</i>_VAL4</td></tr>
<tr><td>5</td><td>VAL5</td></tr>
</table>
0 The corresponding CMPF bit will not cause an interrupt request.
1 The corresponding CMPF bit will cause an interrupt request.
</td></tr>
</table>

40.3.14 Submodule n DMA Enable Register (FlexPWM_SUB*n*_DMAEN)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 1Eh offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	0						VALDE	FAND
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CAPTDE		Reserved				CX1DE	CX0DE
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_DMAEN field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 VALDE	Value Registers DMA Enable This read/write bit enables DMA write requests for the SUB*n*_VAL*x* registers when RF is set. Reset clears VALDE. 0 DMA write requests disabled 1 DMA write requests for the SUB*n*_VAL*x* registers enabled
7 FAND	FIFO Watermark AND Control This read/write bit works in conjunction with the CAPTDE field when it is set to watermark mode (CAPTDE = 01). While the CAxDE, CBxDE, and CXxDE bits determine which FIFO watermarks the DMA read request is sensitive to, this bit determines if the selected watermarks are AND'ed together or OR'ed together in order to create the request.

Table continues on the next page...

FlexPWM_SUB*n*_DMAEN field descriptions (continued)

Field	Description
	0 Selected FIFO watermarks are OR'ed together. 1 Selected FIFO watermarks are AND'ed together.
8–9 CAPTDE	Capture DMA Enable Source Select These read/write bits select the source of enabling the DMA read requests for the capture FIFOs. Reset clears these bits. 00 Read DMA requests disabled. 01 Exceeding a FIFO watermark sets the DMA read request. This requires at least 1 of the CA1DE, CA0DE, CB1DE, CB0DE, CX1DE, or CX0DE bits to also be set in order to determine which watermark(s) the DMA request is sensitive to. 10 A local sync (VAL1 matches counter) sets the read DMA request. 11 A local reload (RF being set) sets the read DMA request.
10–13 Reserved	This field is reserved.
14 CX1DE	Capture X1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X1 FIFO data when CFX1 is set. Reset clears CX1DE. Do not set both this bit and the CX1IE bit.
15 CX0DE	Capture X0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X0 FIFO data when CFX0 is set. Reset clears CX0DE. Do not set both this bit and the CX0IE bit.

40.3.15 Submodule n Output Trigger Control Register (FlexPWM_SUB*n*_TCTRL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 20h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0										OUT_TRIG_EN					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_TCTRL field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 OUT_TRIG_EN	Output Trigger Enables

Table continues on the next page...

FlexPWM_SUB*n*_TCTRL field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>These bits enable the generation of OUT_TRIG0 and OUT_TRIG1 outputs based on the counter value matching the value in one or more of the SUB<i>n</i>_VAL<i>x</i> registers. VAL0, VAL2, and VAL4 are used to generate OUT_TRIG0 and VAL1, VAL3, and VAL5 are used to generate OUT_TRIG1. The OUT_TRIGx signals are only asserted as long as the counter value matches the VAL<i>x</i> value, therefore up to six triggers can be generated (three each on OUT_TRIG0 and OUT_TRIG1) per PWM cycle per submodule.
<table>
<tr><th>OUT_TRIG_EN bit</th><th>Value register</th></tr>
<tr><td>0</td><td>SUB<i>n</i>_VAL0</td></tr>
<tr><td>1</td><td>SUB<i>n</i>_VAL1</td></tr>
<tr><td>2</td><td>SUB<i>n</i>_VAL2</td></tr>
<tr><td>3</td><td>SUB<i>n</i>_VAL3</td></tr>
<tr><td>4</td><td>SUB<i>n</i>_VAL4</td></tr>
<tr><td>5</td><td>SUB<i>n</i>_VAL5</td></tr>
</table>
0 OUT_TRIGx will not set when the counter value matches the VAL<i>x</i> value.
1 OUT_TRIGx will set when the counter value matches the VAL<i>x</i> value.</td></tr>
</table>

40.3.16 Submodule n Fault Disable Mapping Register (FlexPWM_SUB*n*_DISMAP)

This register determines which PWM pins are disabled by the fault protection inputs, illustrated in the table in Fault Protection . Reset sets all of the bits in the fault disable mapping register.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 22h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	1				DISX				DISB				DISA			
Write																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FlexPWM_SUB*n*_DISMAP field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
4–7 DISX	PWMX Fault Disable Mask

Table continues on the next page...

FlexPWM_SUB*n*_DISMAP field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>Each of the four bit of this read/write field is one-to-one associated with the four FAULT<i>x</i> inputs. The PWMX output will be turned off if there is a logic 1 on a FAULT<i>x</i> input and a 1 in the corresponding bit of the DISX field. A reset sets all DISX bits.
<table>
<tr><th>DISX bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table></td></tr>
<tr><td>8–11
DISB</td><td>PWMB Fault Disable Mask
Each of the four bit of this read/write field is one-to-one associated with the four FAULT<i>x</i> inputs. The PWMB output will be turned off if there is a logic 1 on a FAULT<i>x</i> input and a 1 in the corresponding bit of the DISB field. A reset sets all DISB bits.
<table>
<tr><th>DISB bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table></td></tr>
<tr><td>12–15
DISA</td><td>PWMA Fault Disable Mask
Each of the four bit of this read/write field is one-to-one associated with the four FAULT<i>x</i> inputs. The PWMA output will be turned off if there is a logic 1 on a FAULT<i>x</i> input and a 1 in the corresponding bit of the DISA field. A reset sets all DISA bits.
<table>
<tr><th>DISA bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table></td></tr>
</table>

40.3.17 Submodule n Deadtime Count Register 0 (FlexPWM_SUB*n*_DTCNT0)

Deadtime operation is only applicable to complementary channel operation. The 11-bit value written to this register is in terms of peripheral clock cycles regardless of the setting of PRSC and/or CLK_SEL. Reset sets the deadtime count register to a default value of 0x07FF, selecting a deadtime of 2047 clock cycles. This register is not byte accessible.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 24h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					DTCNT0										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

FlexPWM_SUB*n*_DTCNT0 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 DTCNT0	Controls the deadtime during 0 to 1 transitions of the PWMA output (assuming normal polarity) in terms of clock cycles regardless of the setting of PRSC and/or CLK_SEL

40.3.18 Submodule n Deadtime Count Register 1 (FlexPWM_SUB*n*_DTCNT1)

Deadtime operation is only applicable to complementary channel operation. The 11-bit value written to this register is in terms of peripheral clock cycles regardless of the setting of PRSC and/or CLK_SEL. Reset sets the deadtime count register to a default value of 0x07FF, selecting a deadtime of 2047 clock cycles. This register is not byte accessible.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 26h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					DTCNT1										
Write																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

FlexPWM_SUB*n*_DTCNT1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 DTCNT1	Controls the deadtime during 0 to 1 transitions of the complementary PWMB output, in terms of clock cycles regardless of the setting of PRSC and/or CLK_SEL

40.3.19 Submodule n Capture Control X Register (FlexPWM_SUB*n*_CAPTCTRLX)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 30h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	CX1CNT			CX0CNT			CFXWM	
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	EDGCNTX_EN	INPSELX	EDGX1		EDGX0		ONESHOTX	ARMX
Reset	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CAPTCTRLX field descriptions

Field	Description
0–2 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO.
3–5 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO.
6–7 CFXWM	Capture X FIFOs Watermark This field represents the watermark level for capture X FIFOs. The capture flags, CFX1 and CFX0, won't be set until the word count of the corresponding FIFO is greater than this watermark level.
8 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWMX input signal. 0 Edge counter disabled and held in reset. 1 Edge counter enabled.
9 INPSELX	Input Select X This bit selects between the raw PWMX input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. **NOTE:** When INPSELX = 1, the internal edge counter is enabled and the rising and/or falling edges specified by the EDGX0 and EDGX1 fields are ignored. The software must still place a value other than 00 in either or both of the EDGX0 and/or EDGX1 fields in order to enable one or both of the capture registers. 0 Raw PWMX input signal selected as source. 1 Output of edge counter/compare selected as source.

Table continues on the next page...

FlexPWM_SUB*n*_CAPTCTRLX field descriptions (continued)

Field	Description
10–11 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled. 01 Capture falling edges. 10 Capture rising edges. 11 Capture any edge.
12–13 EDGX0	Edge X 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled. 01 Capture falling edges. 10 Capture rising edges. 11 Capture any edge.
14 ONESHOTX	One Shot Mode Aux This bit selects between free running and one shot mode for the input capture circuitry. 0 Free running mode is selected If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMX bit is cleared. No further captures will be performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMX bit is then cleared.
15 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s). 0 Input capture operation is disabled. 1 Input capture operation as specified by the EDGX bits is enabled.

40.3.20 Submodule n Capture Compare X Register (FlexPWM_SUB*n*_CAPTCMPX)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 32h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	EDGCNTX								EDGCMPX							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CAPTCMPX field descriptions

Field	Description
0–7 EDGCNTX	Edge Counter X This read only field contains the edge counter value for the PWMX input capture circuitry.
8–15 EDGCMPX	Edge Compare X This read/write field is the compare value associated with the edge counter for the PWMX input capture circuitry.

40.3.21 Submodule n Capture Value 0 Register (FlexPWM_SUB*n*_CVAL0)

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGX0 bits. This is actually a 4-deep FIFO and not a single register. This register is not byte accessible.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 34h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CAPTVAL0															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CVAL0 field descriptions

Field	Description
0–15 CAPTVAL0	Value captured from the submodule counter

40.3.22 Submodule n Capture Value 0 Cycle Register (FlexPWM_SUB*n*_CVAL0CYC)

This read only register stores the cycle number corresponding to the value captured in CVAL0. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. This is actually a 4 deep FIFO and not a single register.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 36h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0													CVAL0CYC		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CVAL0CYC field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 CVAL0CYC	CVAL0 Cycle Cycle number corresponding to the value captured in CVAL0

40.3.23 Submodule n Capture Value 1 Register (FlexPWM_SUB*n*_CVAL1)

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGX1 bits. This is actually a 4 deep FIFO and not a single register. This register is not byte accessible.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 38h offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CAPTVAL1															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CVAL1 field descriptions

Field	Description
0–15 CAPTVAL1	Value captured from the submodule counter

40.3.24 Submodule n Capture Value 1 Cycle Register (FlexPWM_SUB*n*_CVAL1CYC)

This read only register stores the cycle number corresponding to the value captured in CVAL1. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. This is actually a 4 deep FIFO and not a single register.

Access:

- Supervisor read-only
- User read-only

Address: 0h base + 3Ah offset + (80d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0													CVAL1CYC		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_SUB*n*_CVAL1CYC field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 CVAL1CYC	CVAL1 Cycle Cycle number corresponding to the value captured in CVAL1

40.3.25 Output Enable Register (FlexPWM_OUTEN)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 140h offset = 140h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0				PWMA_EN				PWMB_EN				PWMX_EN			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_OUTEN field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–3
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>4–7
PWMA_EN</td><td>PWMA Output Enables
These bits enable the PWMA outputs of each submodule.
<table>
<tr><th>PWMA_EN bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWMA output disabled.
1 PWMA output enabled.</td></tr>
<tr><td>8–11
PWMB_EN</td><td>PWMB Output Enables
These bits enable the PWMB outputs of each submodule.
<table>
<tr><th>PWMB_EN bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWMB output disabled.
1 PWMB output enabled.</td></tr>
<tr><td>12–15
PWMX_EN</td><td>PWMX Output Enables
These bits enable the PWMX outputs of each submodule. These bits should be set to 0 (output disabled) when a PWMX pin is being used for input capture or deadtime correction.
<table>
<tr><th>PWMX_EN bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWMX output disabled.
1 PWMX output enabled.</td></tr>
</table>

40.3.26 Mask Register (FlexPWM_MASK)

NOTE

> The MASKx bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to the figure in Force Out Logic to see how FORCE_OUT is generated. Reading the MASK bits reads the buffered value and not necessarily the value currently in effect. This double buffering can be overridden by setting the UPDATE_MASK bits.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 142h offset = 142h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0				MASKA				MASKB				MASKX			
Write	UPDATE_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_MASK field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–3
UPDATE_MASK</td><td>Update Mask
These self-clearing bits force the MASK* bits to be immediately updated within each of the submodules without waiting for a FORCE_OUT event. Software may write to any or all of these bits and may set these bits in the same write operation that updates the MASKA, MASKB, and MASKX fields of this register.
0 Normal operation. MASK* bits within the submodules aren’t updated until a FORCE_OUT event occurs within the submodule.
1 Immediate operation. MASK* bits within the corresponding submodules are updated on the following clock edge after setting this bit.</td></tr>
<tr><td>4–7
MASKA</td><td>PWMA Masks
These bits mask the PWMA outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity.
<table>
<tr><th>MASKA bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

FlexPWM_MASK field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>0 PWMA output normal.
1 PWMA output masked.</td></tr>
<tr><td>8–11
MASKB</td><td>PWMB Masks
These bits mask the PWMB outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity.
<table>
<tr><th>MASKB bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWMB output normal.
1 PWMB output masked.</td></tr>
<tr><td>12–15
MASKX</td><td>PWMX Masks
These bits mask the PWMX outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity.
<table>
<tr><th>MASKX bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 PWMX output normal.
1 PWMX output masked.</td></tr>
</table>

40.3.27 Software Controlled Output Register (FlexPWM_SWCOUT)

NOTE

These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to the figure in Force Out Logic to see how FORCE_OUT is generated. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 144h offset = 144h

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	OUT23_3	OUT45_3	OUT23_2	OUT45_2	OUT23_1	OUT45_1	OUT23_0	OUT45_0
Reset	0	0	0	0	0	0	0	0

FlexPWM_SWCOUT field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 OUT23_3	Software Controlled Output 23_3 This bit is only used when SEL23 for submodule 3 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23.
9 OUT45_3	Software Controlled Output 45_3 This bit is only used when SEL45 for submodule 3 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45.
10 OUT23_2	Software Controlled Output 23_2 This bit is only used when SEL23 for submodule 2 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23.
11 OUT45_2	Software Controlled Output 45_2 This bit is only used when SEL45 for submodule 2 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45.
12 OUT23_1	Software Controlled Output 23_1 This bit is only used when SEL23 for submodule 1 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23.
13 OUT45_1	Software Controlled Output 45_1 This bit is only used when SEL45 for submodule 1 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule.

Table continues on the next page...

FlexPWM_SWCOUT field descriptions (continued)

Field	Description
	0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45.
14 OUT23_0	Software Controlled Output 23_0 This bit is only used when SEL23 for submodule 0 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23.
15 OUT45_0	Software Controlled Output 45_0 This bit is only used when SEL45 for submodule 0 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45.

40.3.28 Deadtime Source Select Register (FlexPWM_DTSRCSEL)

NOTE

> The deadtime source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to the figure in Force Out Logic to see how FORCE_OUT is generated. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 146h offset = 146h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SEL23_3		SEL45_3		SEL23_2		SEL45_2		SEL23_1		SEL45_1		SEL23_0		SEL45_0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_DTSRCSEL field descriptions

Field	Description
0–1 SEL23_3	PWM23_3 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 3 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_3 signal is used by the deadtime logic. 01 Inverted generated PWM23_3 signal is used by the deadtime logic.

Table continues on the next page...

FlexPWM_DTSRCSEL field descriptions (continued)

Field	Description
	10 OUT23_3 bit is used by the deadtime logic. 11 EXTA[3] signal is used by the deadtime logic.
2–3 SEL45_3	PWM45_3 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 3 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_3 signal is used by the deadtime logic. 01 Inverted generated PWM45_3 signal is used by the deadtime logic. 10 OUT45_3 bit is used by the deadtime logic. 11 EXTB[3] signal is used by the deadtime logic.
4–5 SEL23_2	PWM23_2 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 2 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_2 signal is used by the deadtime logic. 01 Inverted generated PWM23_2 signal is used by the deadtime logic. 10 OUT23_2 bit is used by the deadtime logic. 11 EXTA[2] signal is used by the deadtime logic.
6–7 SEL45_2	PWM45_2 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 2 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_2 signal is used by the deadtime logic. 01 Inverted generated PWM45_2 signal is used by the deadtime logic. 10 OUT45_2 bit is used by the deadtime logic. 11 EXTB[2] signal is used by the deadtime logic.
8–9 SEL23_1	PWM23_1 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 1 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_1 signal is used by the deadtime logic. 01 Inverted generated PWM23_1 signal is used by the deadtime logic. 10 OUT23_1 bit is used by the deadtime logic. 11 EXTA[1] signal is used by the deadtime logic.
10–11 SEL45_1	PWM45_1 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 1 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_1 signal is used by the deadtime logic. 01 Inverted generated PWM45_1 signal is used by the deadtime logic. 10 OUT45_1 bit is used by the deadtime logic. 11 EXTB[1] signal is used by the deadtime logic.
12–13 SEL23_0	PWM23_0 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 0 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_0 signal is used by the deadtime logic.

Table continues on the next page...

FlexPWM_DTSRCSEL field descriptions (continued)

Field	Description
	01 Inverted generated PWM23_0 signal is used by the deadtime logic. 10 OUT23_0 bit is used by the deadtime logic. 11 EXTA[0] signal is used by the deadtime logic.
14–15 SEL45_0	PWM45_0 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 0 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_0 signal is used by the deadtime logic. 01 Inverted generated PWM45_0 signal is used by the deadtime logic. 10 OUT45_0 bit is used by the deadtime logic. 11 EXTB[0] signal is used by the deadtime logic.

40.3.29 Master Control Register (FlexPWM_MCTRL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 148h offset = 148h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	IPOL				RUN				0				LDOK			
Write									CLDOK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_MCTRL field descriptions

Field	Description
0–3 IPOL	Current Polarity This buffered read/write bit is used to select between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output in each submodule. IPOL is ignored in independent mode. IPOL bit / PWM submodule: 0 / 0; 1 / 1; 2 / 2; 3 / 3 **NOTE:** The IPOL bit does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading the IPOL bit reads the buffered value and not necessarily the value currently in effect.

IPOL bit	PWM submodule
0	0
1	1
2	2
3	3

Table continues on the next page...

FlexPWM_MCTRL field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>0 PWM23 (PWM submodule) is used to generate complementary PWM pair.
1 PWM45 (PWM submodule) is used to generate complementary PWM pair.</td></tr>
<tr><td>4–7
RUN</td><td>Run

This read/write bit enables the clocks to the PWM generator in each submodule. When RUN equals zero, the submodule counter is reset. A reset clears RUN.

<table>
<tr><th>RUN bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>

NOTE: For proper initialization of the LDOK and RUN bits, see Initialization .

0 PWM generator disabled
1 PWM generator enabled</td></tr>
<tr><td>8–11
CLDOK</td><td>Clear Load Okay

This write only bit is used to clear the LDOK bit. Write a 1 to this location to clear the corresponding LDOK. If a reload occurs with LDOK set at the same time that CLDOK is written, then the reload will not be performed and LDOK will be cleared. This bit is self clearing and always reads as a 0.

<table>
<tr><th>CLDOK bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
</td></tr>
<tr><td>12–15
LDOK</td><td>Load Okay

This field loads the PRSC bits of SUB*n*_CTRL1, SUB*n*_INIT, and SUB*n*_VAL*x* registers into a set of buffers in each submodule. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if LDMOD is clear or immediately if LDMOD is set. Set LDOK by reading it when it is logic zero and then writing a logic one to it. SUB*n*_VAL*x*, SUB*n*_INIT, and SUB*n*_CTRL*x*[PRSC] cannot be written while LDOK is set. LDOK is automatically cleared after the new values are loaded, or can be manually cleared before a reload by writing a logic 1 to CLDOK. This bit cannot be written with a zero. LDOK can be set in DMA mode when the DMA indicates that it has completed the update of all SUB*n*_INIT, SUB*n*_VAL*x*, or SUB*n*_CTRL*x*[PRSC]> registers. Reset clears LDOK.

<table>
<tr><th>LDOK bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

FlexPWM_MCTRL field descriptions (continued)

Field	Description
	LDOK bit: 3 — PWM submodule: 3 **NOTE:** For proper initialization of the LDOK and RUN bits, see Initialization . 0 Do not load new values. 1 Load prescaler, modulus, and PWM values.

40.3.30 Reserved Register (FlexPWM_RESERVED)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 14Ah offset = 14Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_RESERVED field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.3.30 Fault Control Register (FlexPWM_FCTRL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 14Ch offset = 14Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	FLVL				FAUTO				FSAFE				FIE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_FCTRL field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–3
FLVL</td><td>Fault Level
These read/write bits select the active logic level of the individual fault inputs. A reset clears FLVL.
<table>
<tr><th>FLVL bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 A logic 0 on the fault input indicates a fault condition.
1 A logic 1 on the fault input indicates a fault condition.</td></tr>
<tr><td>4–7
FAUTO</td><td>Automatic Fault Clearing
These read/write bits select automatic or manual clearing of faults. A reset clears FAUTO.
<table>
<tr><th>FAUTO bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 Manual fault clearing. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear at the start of a half cycle or full cycle depending the state of the FFULL bits. This is further controlled by the FSAFE bits.
1 Automatic fault clearing. PWM outputs disabled by this fault are enabled when the FFPINx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits without regard to the state of FFLAGx bit.</td></tr>
<tr><td>8–11
FSAFE</td><td>Fail Safe Mode
These read/write bits select the safety mode during manual fault clearing. A reset clears FSAFE.
<table>
<tr><th>FSAFE bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
<b>NOTE:</b> The FFPINx bit may indicate a fault condition still exists even though the actual fault signal at the FAULT<i>x</i> pin is clear due to the fault filter latency.
0 Normal mode. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits without regard to the state of the FFPINx bit. The PWM outputs disabled by this fault input will not be re-enabled until the actual</td></tr>
</table>

Table continues on the next page...

FlexPWM_FCTRL field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>FAULT<i>x</i> input signal de-asserts since the fault input will combinationally disable the PWM outputs (as programmed in DISMAP).
1 Safe mode. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear and the FFPINx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits.</td></tr>
<tr><td>12–15
FIE</td><td>Fault Interrupt Enables
This read/write bit enables CPU interrupt requests generated by the FAULT<i>x</i> pins. A reset clears FIE.
<table>
<tr><th>FIE bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
<b>NOTE:</b> The fault protection circuit is independent of the FIEx bit and is always active. If a fault is detected, the PWM outputs are disabled according to the disable mapping register.
0 FAULT<i>x</i> CPU interrupt requests disabled.
1 FAULT<i>x</i> CPU interrupt requests enabled.</td></tr>
</table>

40.3.31 Fault Status Register (FlexPWM_FSTS)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 14Eh offset = 14Eh

Bit	0	1	2	3	4	5	6	7
Read	0			FTEST	FFPIN			
Write								
Reset	0	0	0	0	*	*	*	*

Bit	8	9	10	11	12	13	14	15
Read	FFULL				FFLAG			
Write					w1c			
Reset	0	0	0	0	*	*	*	*

* Notes:
- FFLAG field: Undefined
- FFPIN field: Undefined

FlexPWM_FSTS field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–2
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>3
FTEST</td><td>Fault Test
These read/write bit is used to simulate a fault condition. Setting this bit will cause a simulated fault to be sent into all of the fault filters. The condition will propagate to the fault flags and possibly the PWM outputs depending on the DISMAP settings. Clearing this bit removes the simulated fault condition.
0 No fault.
1 Cause a simulated fault.</td></tr>
<tr><td>4–7
FFPIN</td><td>Filtered Fault Pins
These read-only bits reflect the current state of the filtered FAULT<i>x</i> pins converted to high polarity. A logic 1 indicates a fault condition exists on the filtered FAULT<i>x</i> pin. A reset has no effect on FFPIN.
<table>
<tr><th>FFPIN bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table></td></tr>
<tr><td>8–11
FFULL</td><td>Full Cycle
These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition.
0 PWM outputs are re-enabled at the start of a full or half cycle.
1 PWM outputs are re-enabled only at the start of a full cycle.</td></tr>
<tr><td>12–15
FFLAG</td><td>Fault Flags
These read-only flag is set within two CPU cycles after a transition to active on the FAULT<i>x</i> pin. Clear FFLAGx by writing a logic one to it. A reset clears FFLAG.
<table>
<tr><th>FFLAG bit</th><th>PWM submodule</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
0 No fault on the FAULT<i>x</i> pin.
1 Fault on the FAULT<i>x</i> pin.</td></tr>
</table>

40.3.32 Fault Filter Register (FlexPWM_FFILT)

The settings in this register are shared among each of the fault input filters.

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 150h offset = 150h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	GSTR	0				FILT_CNT			FILT_PER							
Write		▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FlexPWM_FFILT field descriptions

Field	Description
0 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 peripheral clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 0 Fault input glitch stretching is disabled. 1 Input fault signals will be stretched to at least 2 peripheral clock cycles.
1–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FILT_CNT affects the input latency as described in Input Filter Considerations .
8–15 FILT_PER	Fault Filter Period These bits represent the sampling period (in peripheral clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency as described in Input Filter Considerations .

40.3.33 Input Filter Considerations

The FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT+3 power.

The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of ((FILT_CNT+4) x FILT_PER x peripheral clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set the FFLAG and FFPIN bits of the FSTS register.

40.4 Functional Description

This section describes the functionality of the FlexPWM.

40.4.1 PWM Capabilities

This section describes some capabilities of the PWM module.

40.4.1.1 Center Aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in following figure.

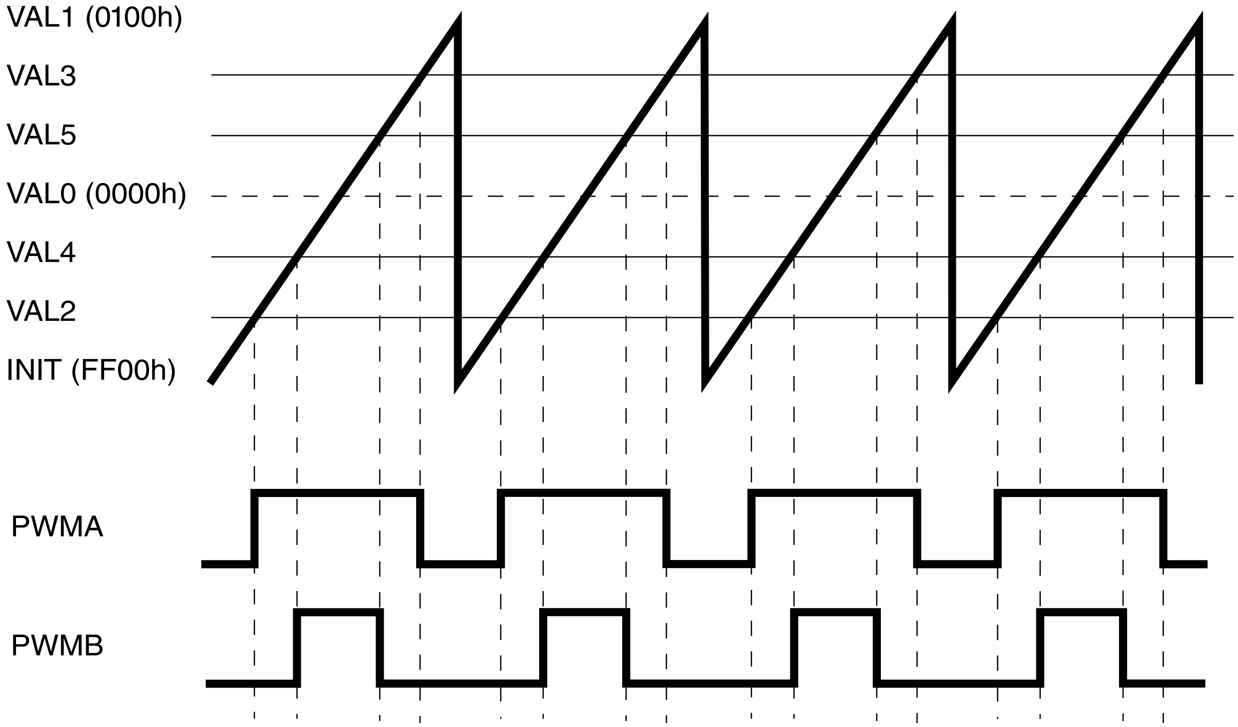


Figure 40-3. Center Aligned Example

The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn on edge and the turn off edge. This double action edge generation not only gives the user control over the pulse width, but over the relative alignment of the signal as well. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn on and turn off edge values.

The figure above also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then the PWM generator operates in "signed" mode. This means that if each PWM's turn on and turn off edge values are also the same number but only different in their sign, the "on" portion of the output signal will be centered around a count value of zero. Therefore, only one PWM value needs to be calculated in software and then this value and its negative are provided to the submodule as the turn off and turn on edges respectively. This technique will result in a pulse width that always consists of an odd number of timer counts. If all PWM signal edge calculations follow this same convention, then the signals will be center aligned with respect to each other, which is the goal. Of course, center alignment between the signals is not restricted to symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

40.4.1.2 Edge Aligned PWMs

When the turn on edge for each pulse is specified to be the INIT value, then edge aligned operation results, as illustrated in the following figure. Therefore, only the turn off edge value needs to be periodically updated to change the pulse width.

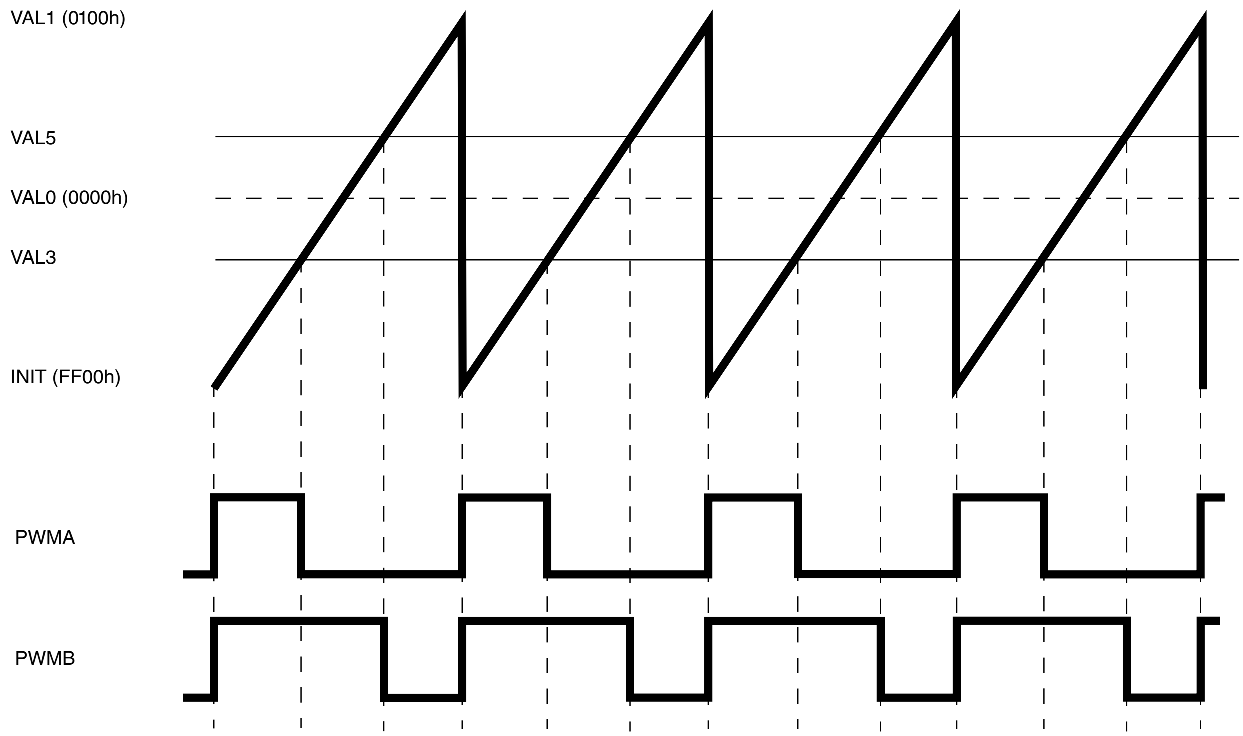


Figure 40-4. Edge Aligned Example (INIT=VAL2=VAL4)

With edge aligned PWMs, another example of the benefits of signed mode can be seen. A common way to drive an H-bridge is to use a technique called "bipolar" PWMs where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% result in negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn off edge value and the motor voltage, INCLUDING the sign. So once again, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

40.4.1.3 Phase Shifted PWMs

In the previous sections, the benefits of signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases ARE applied to the turn on and turn off edges of different PWM signal, the signals will be phase shifted with respect to each other, as illustrated in the following figure. This results in certain advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from the different phases occur at nearly the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does NOT affect the duty cycle so average load voltage is not affected.

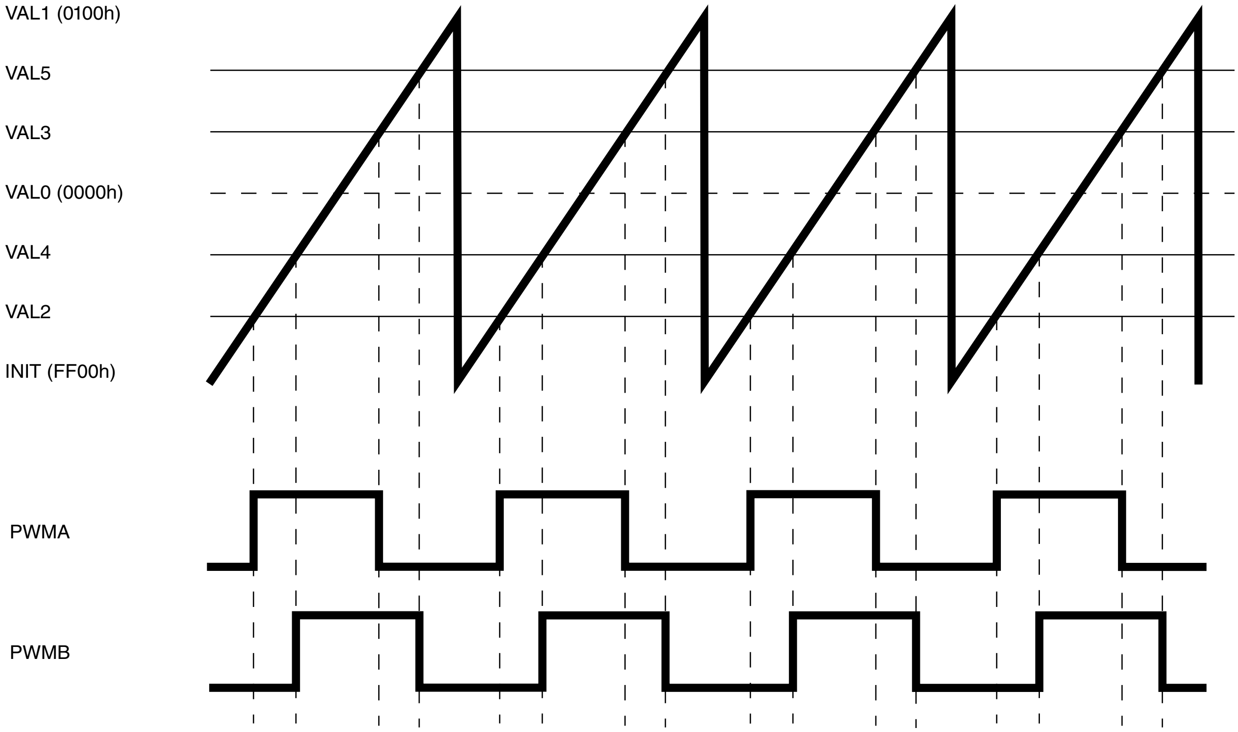


Figure 40-5. Phase Shifted Outputs Example

An additional benefit of phase shifted PWMs can be seen in the following figure. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to always generate a square wave with 50% duty cycle. This works out nicely for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching requirements of the transistors. Notice that the square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% making this technique ideally suited for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

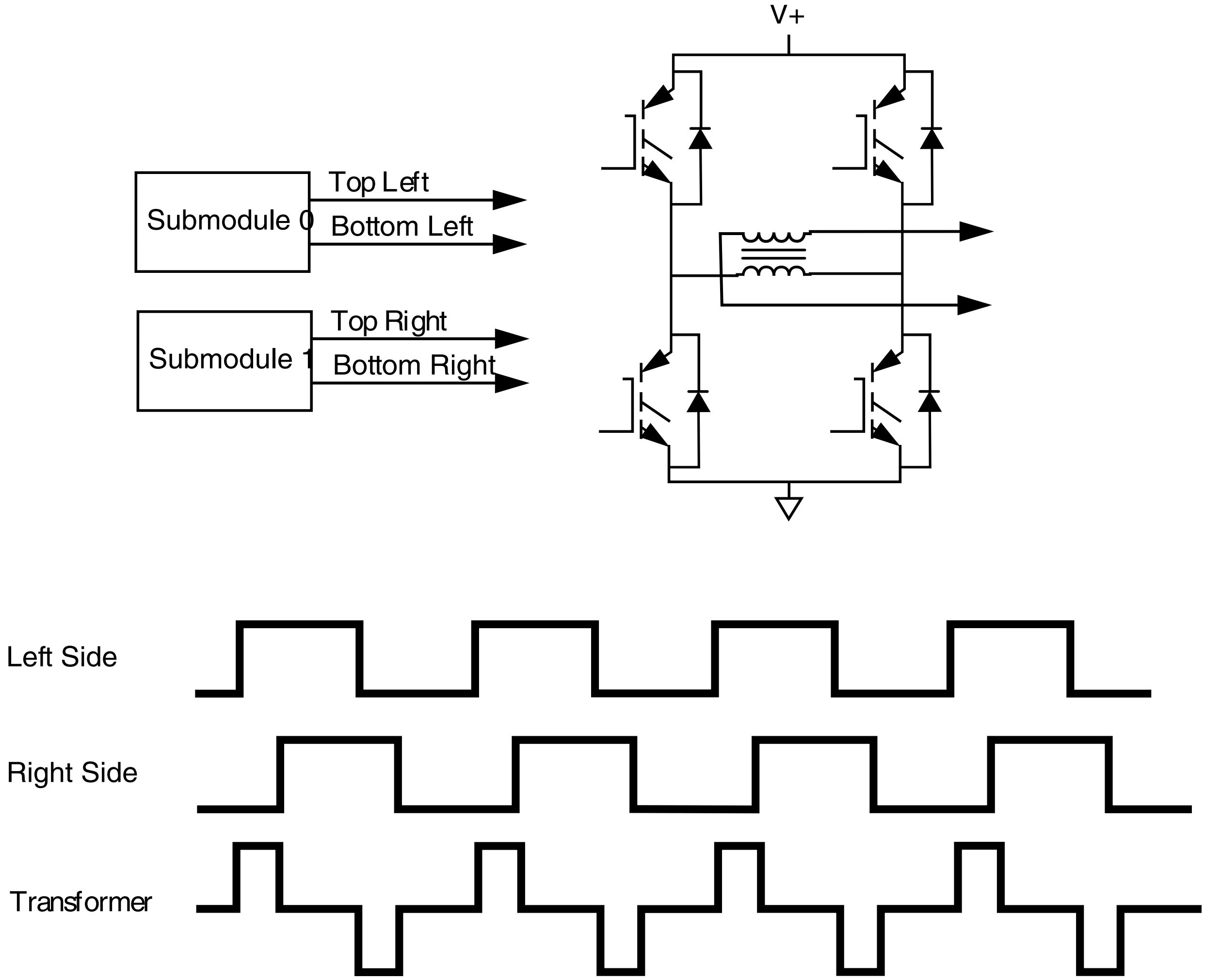


Figure 40-6. Phase Shifted PWMs Applied to a Transformer Primary

40.4.1.4 Double Switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three phase reconstruction. This method support two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labelled as PWMA in the figure) while VAL4 and VAL5 are used to generate the odd channel. The two channels are combined using XOR logic (see Figure 40-17) as shown in the following figure. The DBLPWM signal can be run through the deadtime insertion logic.

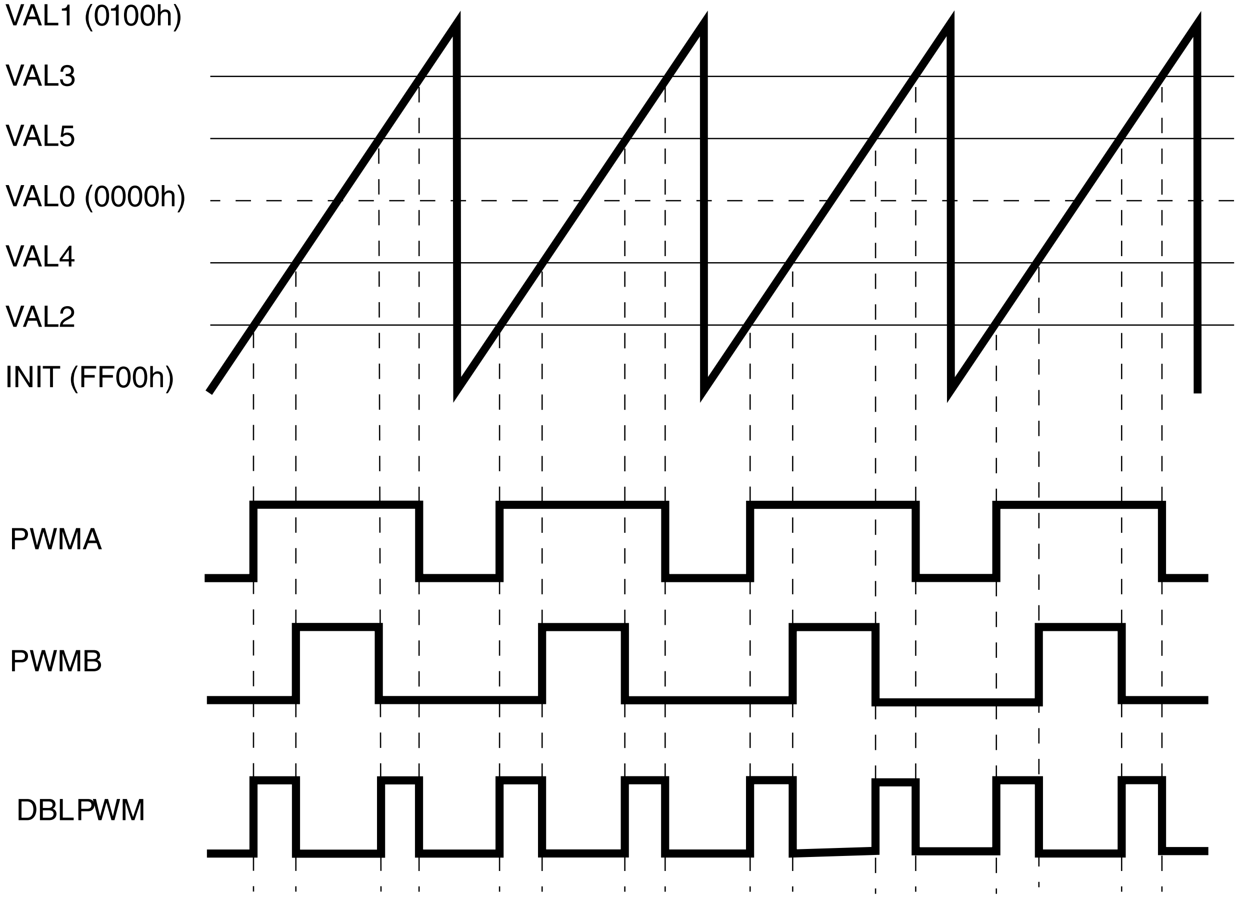


Figure 40-7. Double Switching Output Example

40.4.1.5 ADC Triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. the following figure shows how this is accomplished. When specifying complimentary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators are free to perform other functions. In this example, the software doesn't have to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

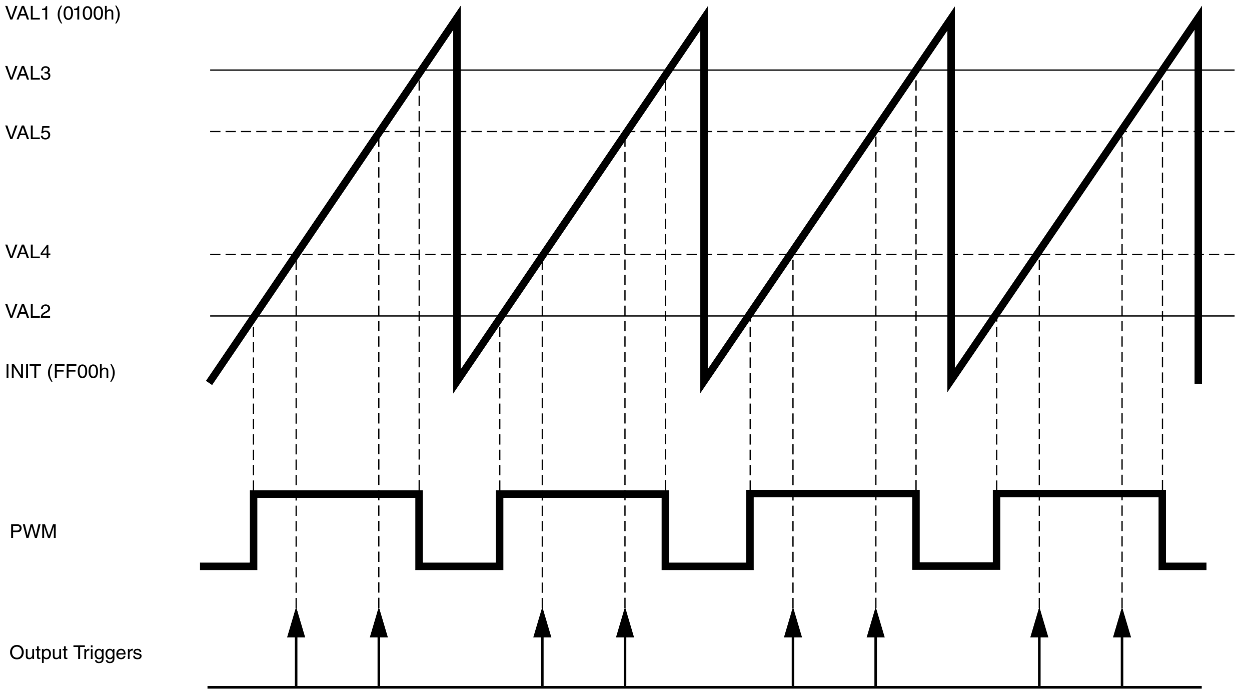


Figure 40-8. Multiple Output Trigger Generation in Hardware

Since each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. The following figure shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. A suggested use for this configuration would be to use the lower frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In this figure, ALL submodule comparators are shown being used for ADC trigger generation.

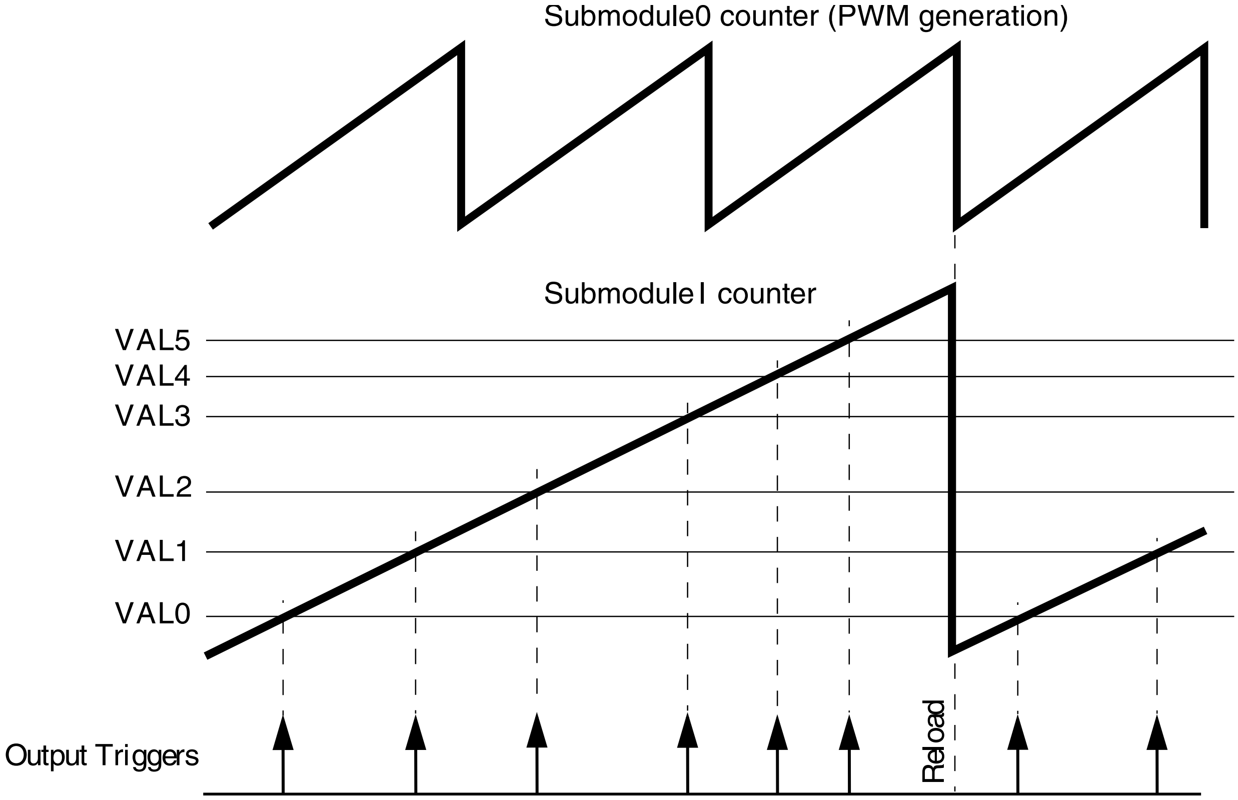


Figure 40-9. Multiple Output Triggers Over Several PWM Cycles

40.4.1.6 Enhanced Capture Capabilities (E-Capture)

When a PWMX pin is not being used for PWM generation, it can be used to perform input captures. Recall that for PWM generation BOTH edges of the PWM signal are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. By simply programming the desired edge of each capture circuit, period and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8 bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature allows the module to count a specified number of edge events and then perform a capture and interrupt. The following figure illustrates some of the functionality of the E-Capture circuit.

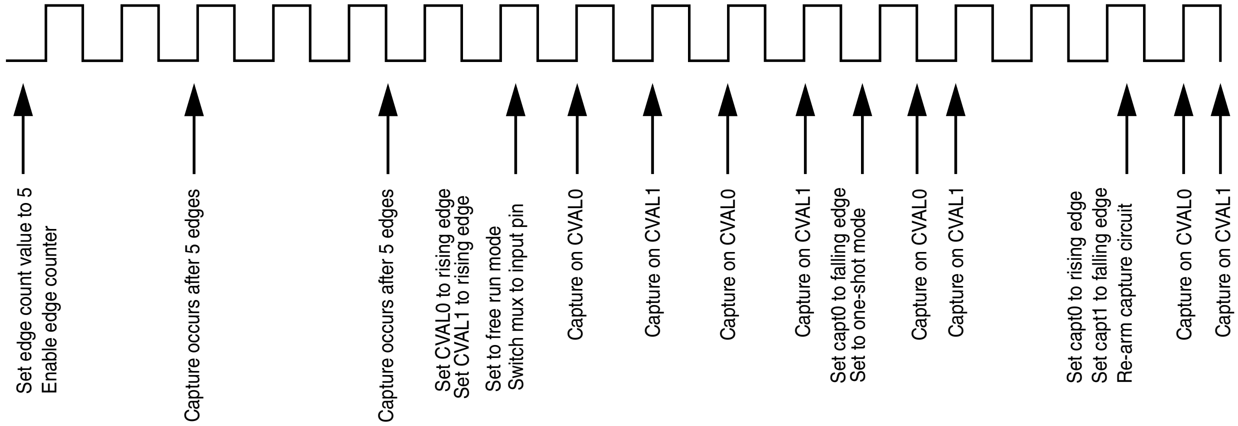


Figure 40-10. Capture Capabilities of the E-Capture Circuit

When a submodule is being used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (e.g, PWMX) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16 bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is perfectly suited for the application. As an example, consider the following figure. In this application the output of a PWM power stage is connected to the PWMX pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This will result in new load pulse width data being acquired every PWM cycle. To calculate the pulse width, simply subtract the CVAL0 register value from the CVAL1 register value. This measurement is extremely beneficial when performing dead-time distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays.

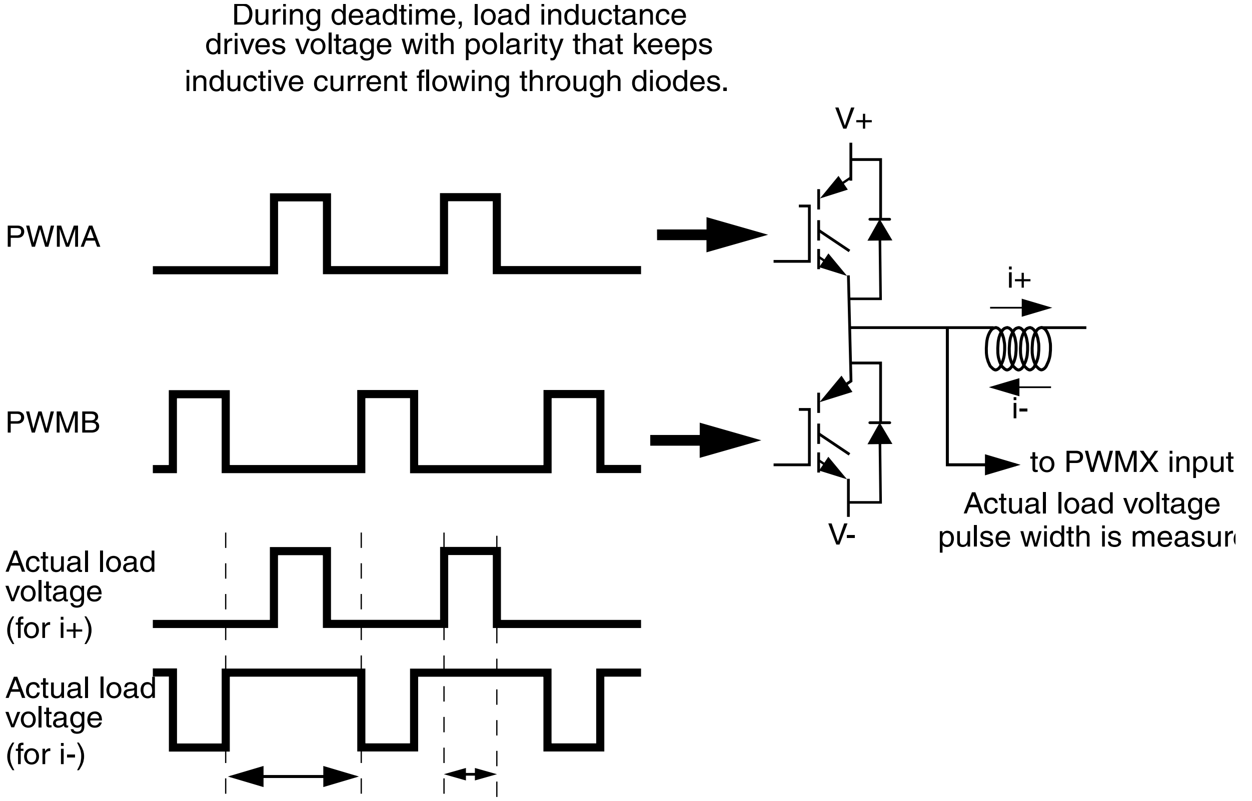


Figure 40-11. Output Pulse Width Measurement Possible with the E-Capture Circuit

40.4.1.7 Synchronous Switching of Multiple Outputs

Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. This feature can be extremely useful in commutated motor applications where the next commutation state can be laid in ahead of time and then immediately switched to the outputs when the appropriate condition or time is reached. Not only do all the changes occur synchronously on all submodule outputs, but they occur IMMEDIATELY after the trigger event occurs eliminating any interrupt latency.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module and, in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event using the SWCOUT and DTSRCSEL registers. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that might occur do not violate deadtime on the power stage when in complementary mode.

The following figure shows a popular application that can benefit from this feature. On a brushless DC motor it is desirable on many cases to spin the motor without need of hall-effect sensor feedback. Instead, the back EMF of the motor phases is monitored and this information is used to schedule the next commutation event. The top waveforms of the following figure are a simplistic representation of these back EMF signals. Timer compare events (represented by the long vertical lines in the diagram) are scheduled based on the zero crossings of the back-EMF waveforms. The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event. When it happens, the output compare of the timer drives the FORCE_OUT signal which immediately changes the state of the PWM pins to the next commutation state with no software latency.

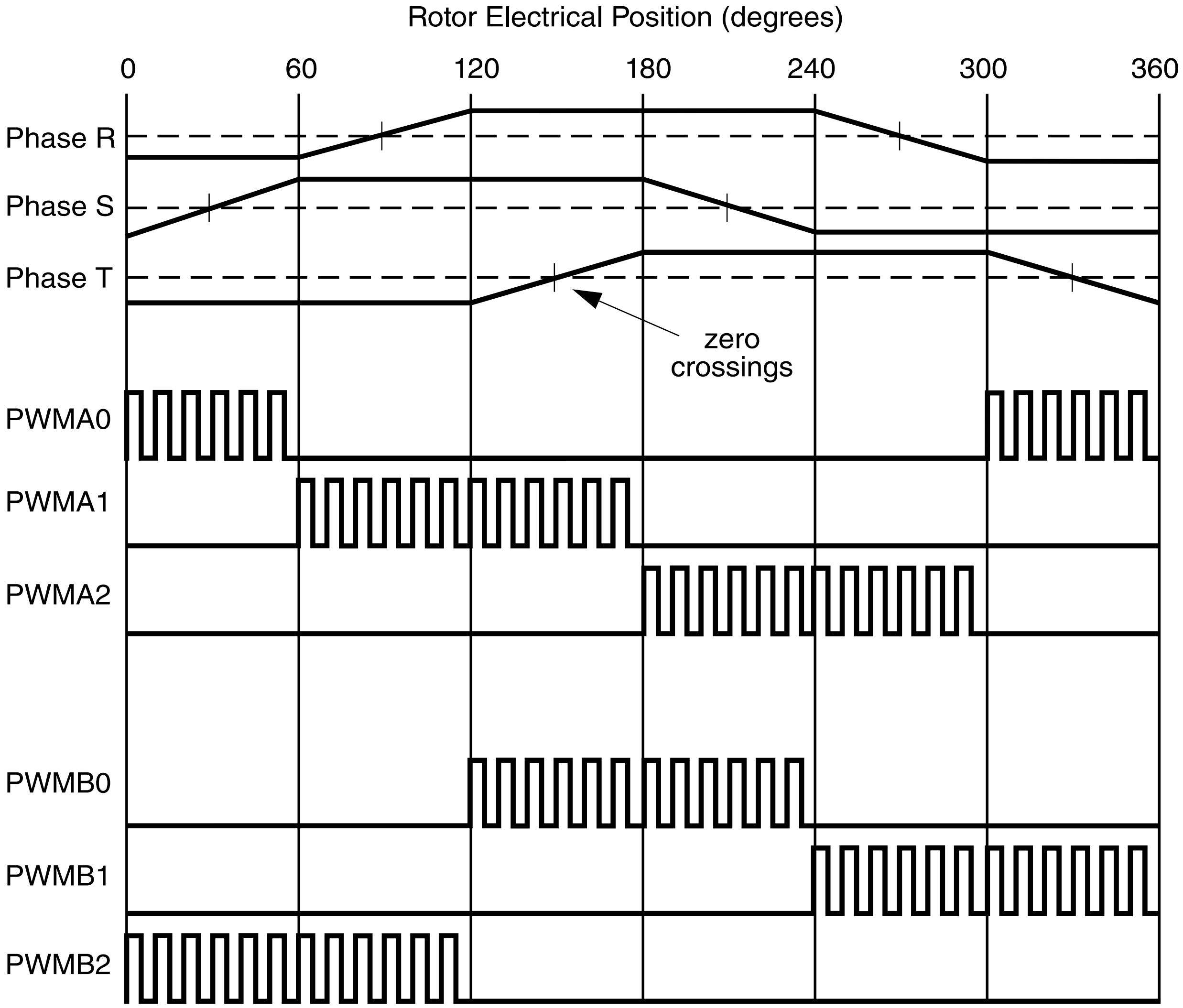


Figure 40-12. Sensorless BLDC Commutation Using the Force Out Function

40.4.1.8 Phase shifted full bridge for DC/DC converters

Use CTRL1[PHSSHFT] to allow the EXT_SWITCH input to control the output of PWMA0/PWMB0. The EXT_SWITCH input switches the source of the submodule0 output from the SUB0_VAL*x* registers to the SUB1_VAL*x* registers. This muxing can be seen in Figure 40-16.

40.4.2 Functional Details

This section describes the implementation of various features of the PWM in greater detail.

40.4.2.1 PWM Clocking

The following figure shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the peripheral clock, EXT_CLK, and AUX_CLK. The EXT_CLK is generated by an on-chip resource such as a Timer module and goes to all of the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8 bit prescaler and RUN bit from submodule0 can control all of the submodules. When AUX_CLK is selected as the source for the submodule clock, the RUN bit from submodule0 is used instead of the local RUN bit from this submodule.

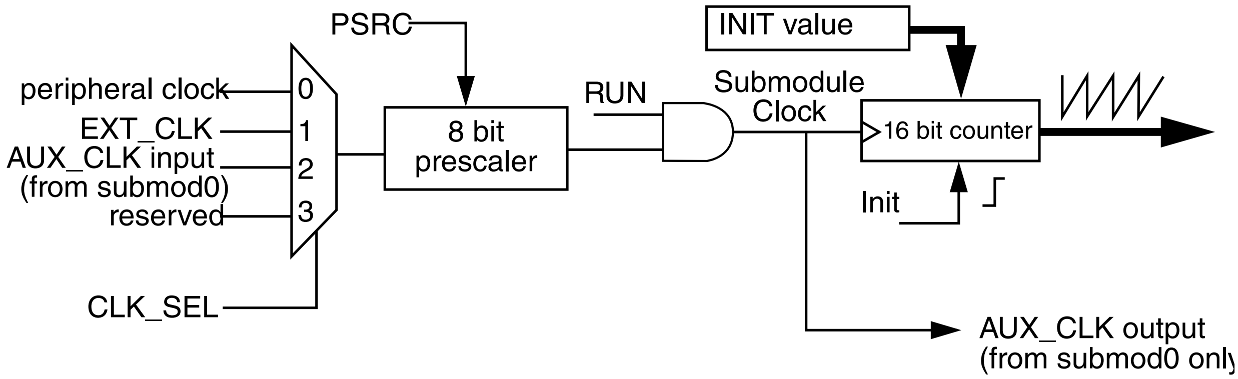


Figure 40-13. Clocking Block Diagram for Each PWM Submodule

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the peripheral clock frequency by 1-128. The prescaler bits, PRSC, in the control register (CTRL1), select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until the LDOK bit is set and a new PWM reload cycle begins or LDMOD is set.

40.4.2.2 Register Reload Logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using the LDFQ bits and the FULL bit. A half cycle reload option is also supported (HALF) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the SUB*n*VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As illustrated in the following figure, the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

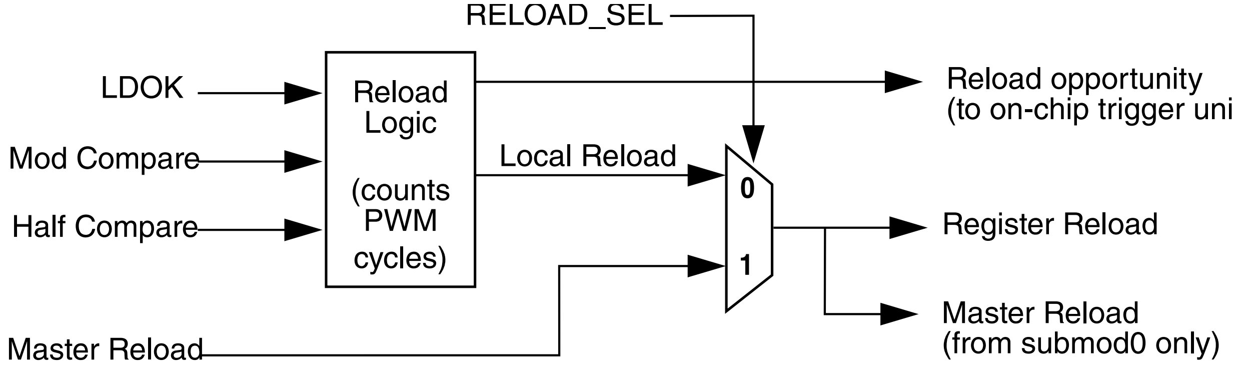

RELOAD_SEL
LDOK
Mod Compare
Half Compare
Master Reload
Reload Logic (counts PWM cycles)
Local Reload
0
1
Reload opportunity (to on-chip trigger uni
Register Reload
Master Reload (from submod0 only)

Figure 40-14. Register Reload Logic

40.4.2.3 Counter Synchronization

Referring to the following figure, the 16 bit counter will count up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Sync signal which is one of four possible sources used to cause the 16 bit counter to be initialized with INIT. If Local Sync is selected as the counter initialization signal, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level.

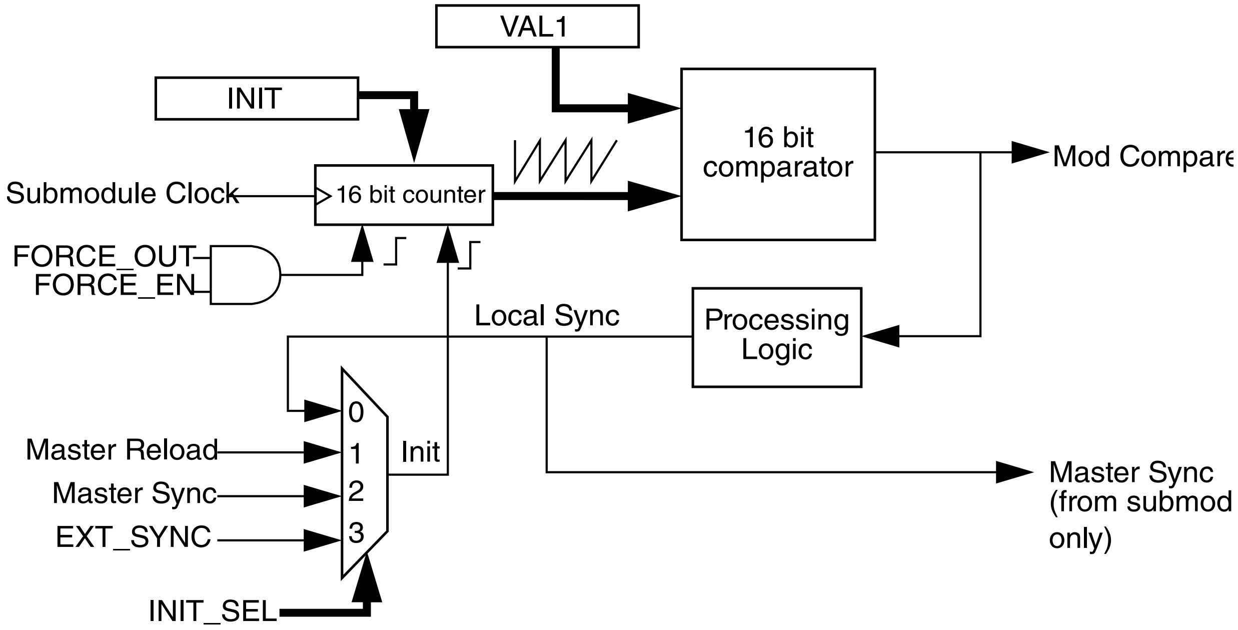


Figure 40-15. Submodule Timer Synchronization

The Master Sync signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0. The SUB*n*_VAL1 register and associated comparator of the other submodules can then be freed up for other functions such as PWM generation, input captures, output compares, or output triggers.

The EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is usually commensurate to the sampling frequency of the software control algorithm, the submodule counter period will therefore equal the sampling period. As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that the FORCE_EN bit is set. As indicated by the figure above, this constitutes a second init input into the counter which will cause the counter to initialize regardless of which signal is selected as the counter init signal. The FORCE_OUT signal is provided mainly for commutated applications. When PWM signals are commutated on an inverter controlling a brushless DC motor, it is necessary to restart the PWM cycle at the beginning of the commutation interval. This action effectively resynchronizes the PWM waveform to the commutation timing. Otherwise, the average voltage applied to a motor winding integrated over the entire commutation interval will be a function of the timing between the asynchronous commutation event with respect to the PWM cycle. The effect is more critical at higher motor speeds where each commutation interval may consist of only a few PWM cycles. If the counter is not initialized at the start of each commutation interval, the result will be an oscillation caused by the beating between the PWM frequency and the commutation frequency.

40.4.2.4 PWM Generation

The following figure illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated SUB*n*_VAL*x* registers are utilized for each PWM output signal. One comparator and VALx register is used to control the turn on edge while a second comparator and SUB*n*_VAL*x* register control the turn off edge.

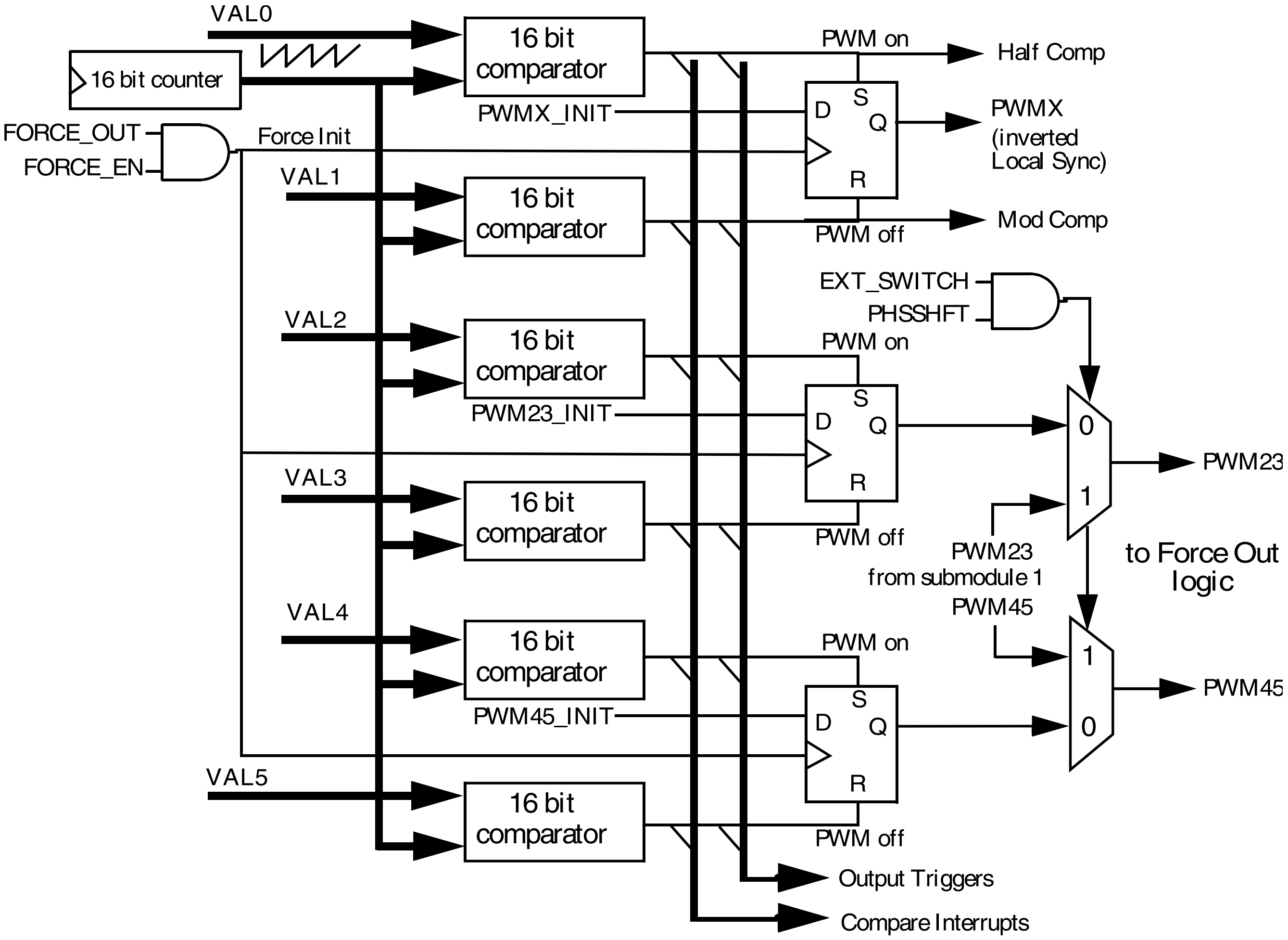


Figure 40-16. PWM Generation Hardware

The generation of the Local Sync signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a rising edge of the Local Sync signal, comparator 1 generates a falling edge. Comparator 1 is also hardwired to the reload logic to generate the half cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the SUB*n*_VAL1 register minus the INIT value, then the half cycle reload pulse will occur exactly half way through the timer count period and the Local Sync will have a 50% duty cycle. On the other hand, if the SUB*n*_VAL1 and SUB*n*_VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the Local Sync signal effectively turning it into an auxiliary PWM signal (PWMX) assuming that the PWMX pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Sync signal, each submodule is capable of generating 3 PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can also be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The muxes for PWM23 and PWM45, shown in the figure above, are only in submodule0.

If the comparators assert both the set and reset of the flip-flop at the same time, then the flop output goes to 0.

40.4.2.5 Output Compare Capabilities

By using the SUB*n*_VAL*x* registers in conjunction with the submodule timer and 16 bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high
- An output compare sets the output low
- An output compare generates an interrupt
- An output compare generates an output trigger

Referring again to Figure 40-16, an output compare is initiated by programming a SUB*n*_VAL*x* register for a timer compare which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWMA signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the SUB*n*_VAL3 register must be programmed to a value outside of the modulus range of the counter. Therefore, a compare that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWMA signal that sets it low, the SUB*n*_VAL3 register is programmed with the appropriate count value and the SUB*n*_VAL2 register is programmed with a value outside the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt or output trigger can be generated when the compare event occurs.

40.4.2.6 Force Out Logic

For each submodule software can select between seven signal sources for the FORCE_OUT signal: the local FORCE bit, the Master Force signal from submodule0, the local Reload signal, the Master Reload signal from submodule0, the Local Sync signal, the Master Sync signal from submodule0, or the EXT_FORCE signal from on or off chip depending on the chip architecture. The local signals are used when the user simply wants to change the signals on the output pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master signals or EXT_FORCE signal should be selected.

The following figure illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the EXTA or EXTB alternate external control signals. The selection can be determined ahead of time and, when a FORCE_OUT event occurs, these values are presented to the signal selection mux which immediately switches the requested signal to the output of the mux for further processing downstream.

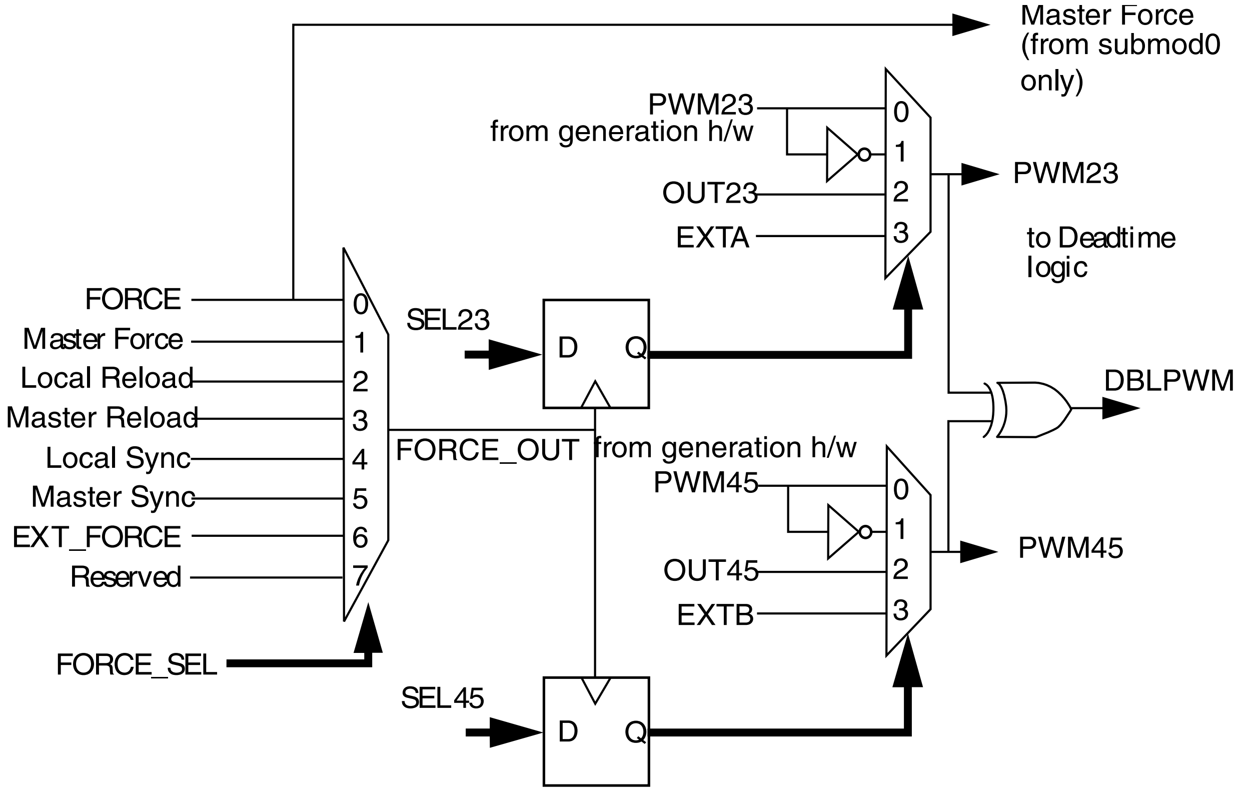


Figure 40-17. Force Out Logic

The local FORCE signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the local FORCE bit of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

40.4.2.7 Independent or Complementary Channel Operation

Writing a logic one to the INDEP bit of the CNFG register configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

Writing a logic zero to the INDEP bit configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in the following figure in complementary channel operation. Which signal is connected to the output pin (PWM23 or PWM45) is determined by the IPOL bit.

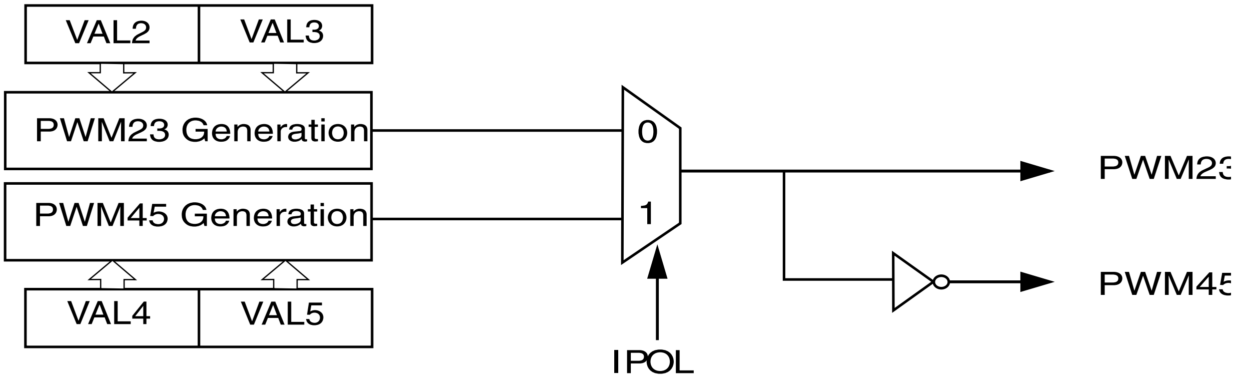


Figure 40-18. Complementary Channel Pair

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, such as the one in the following figure.

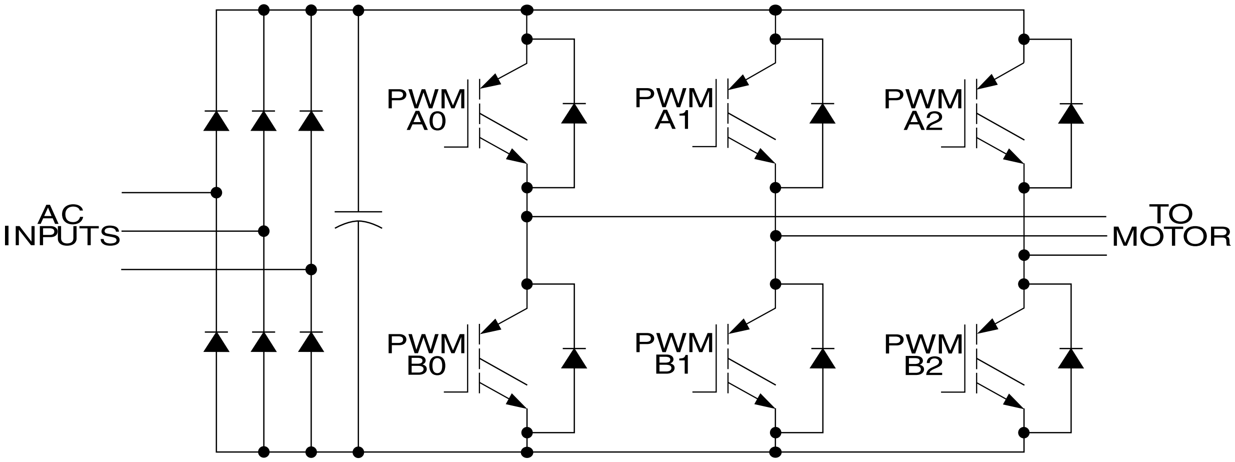


Figure 40-19. Typical 3 Phase AC Motor Drive

Complementary operation allows the use of the deadtime insertion feature.

40.4.2.8 Deadtime Insertion Logic

The following figure shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

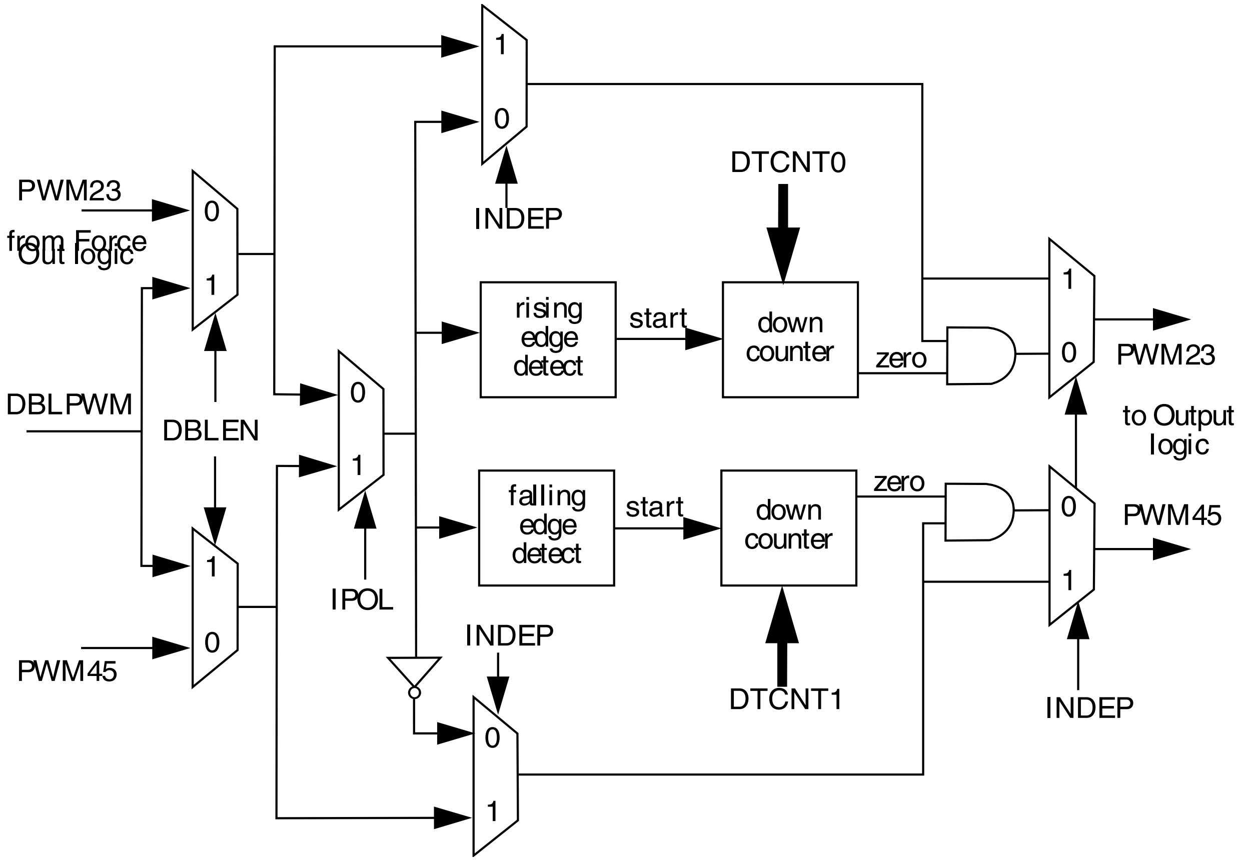


Figure 40-20. Deadtime Insertion and Fine Control Logic

While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in the figure above. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

Note

> To avoid short circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between top and bottom transistor. However, the transistor's characteristics may cause its switching-off time to be longer than its switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as illustrated in the following figure.

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of peripheral clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

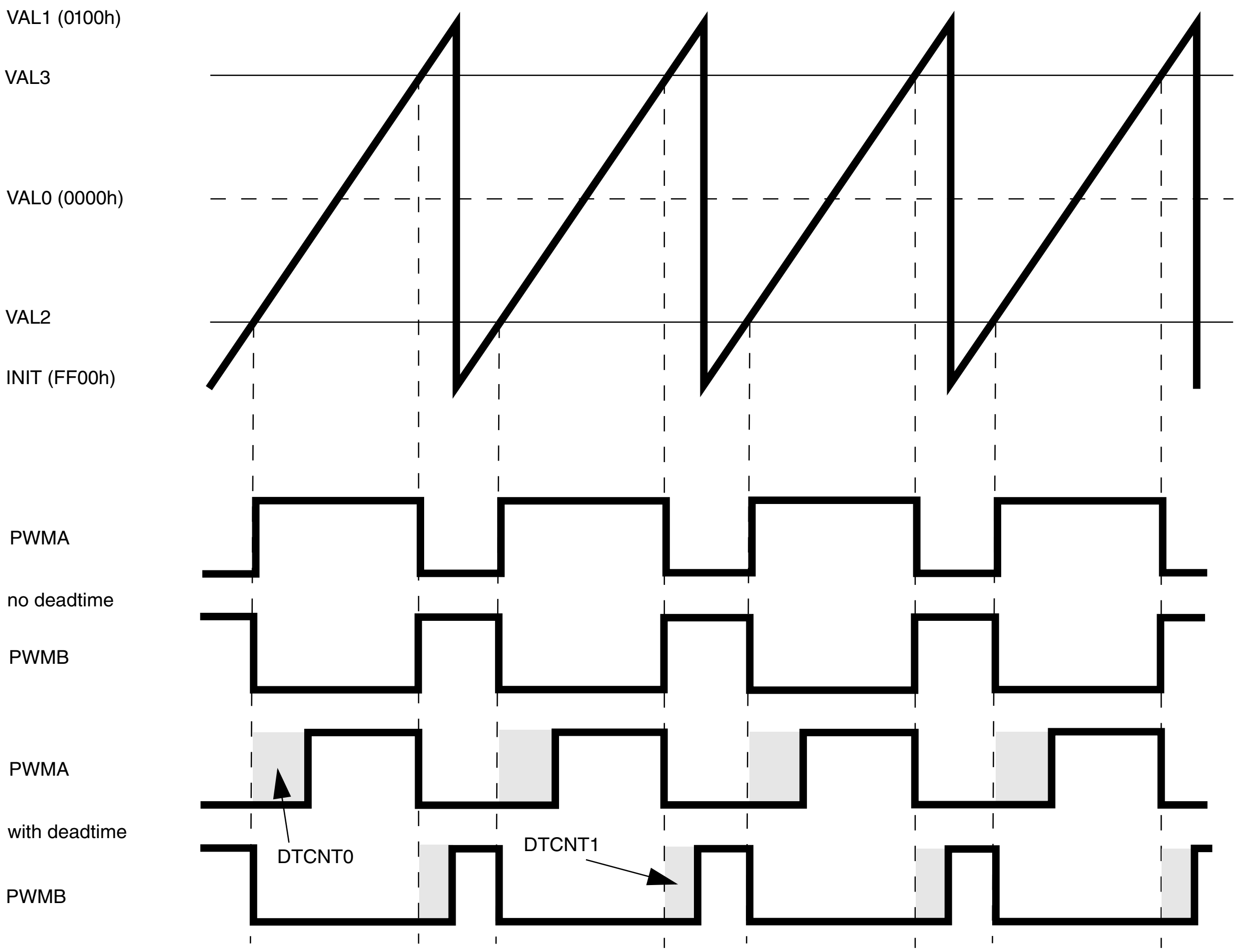


Figure 40-21. Deadtime Insertion

40.4.2.8.1 Top/Bottom Correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introduce distortion in the output voltage. See the following figure. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

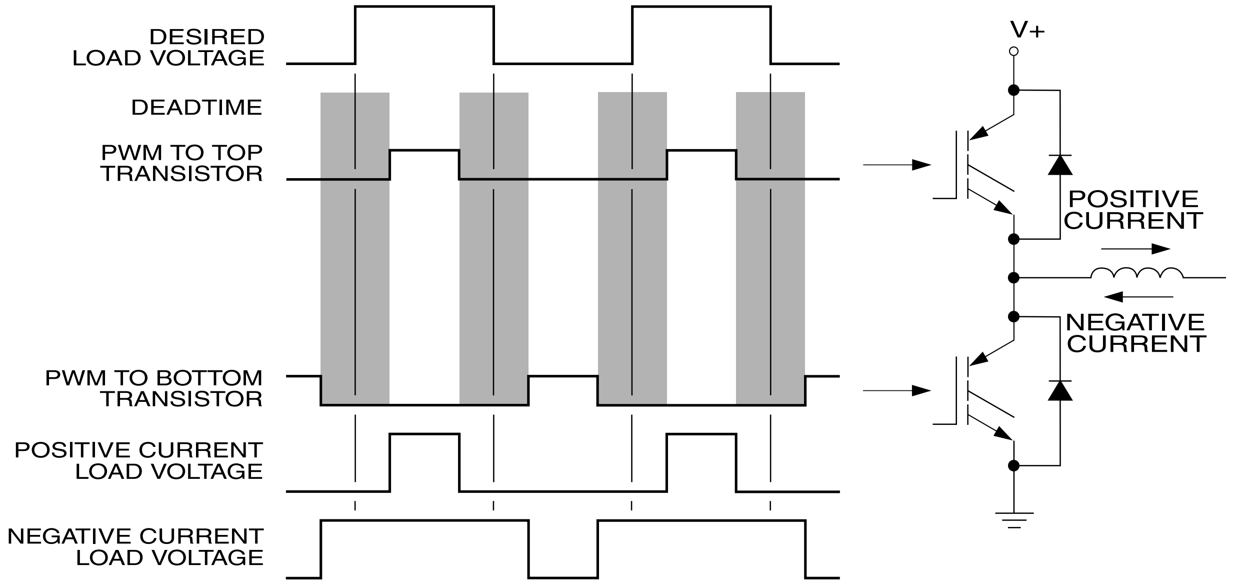


Figure 40-22. Deadtime Distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths where shortened by deadtime insertion, the averaged sinusoidal output will be less than the desired value. However, when deadtime is inserted, it creates a distortion in the motor current waveform. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the motor current for that pair. See the figure above. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the SUB*n*_VAL*x* registers. Either the SUB*n*_VAL2/SUB*n*_VAL3 or the SUB*n*_VAL4/ SUB*n*_VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the SUB*n*_VAL2/SUB*n*_VAL3 or SUB*n*_VAL4/SUB*n*_VAL5 pair is active depends on either:

- The state of the current status pin, PWMx, for that driver
- The state of the odd/even correction bit, IPOL, for that driver

To correct deadtime distortion, software can decrease or increase the value in the appropriate SUB*n*_VAL*x* register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

40.4.2.8.2 Manual Correction

To detect the current status, the voltage on each PWMx pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in the DTx bits in the CTRL1 register. The DTx bits are a timing marker especially indicating when to toggle between PWM value registers. Software can then set the IPOL bit to switch between SUB*n*_VAL2/SUB*n*_VAL3 and SUB*n*_VAL4/SUB*n*_VAL5 register pairs according to DTx values.

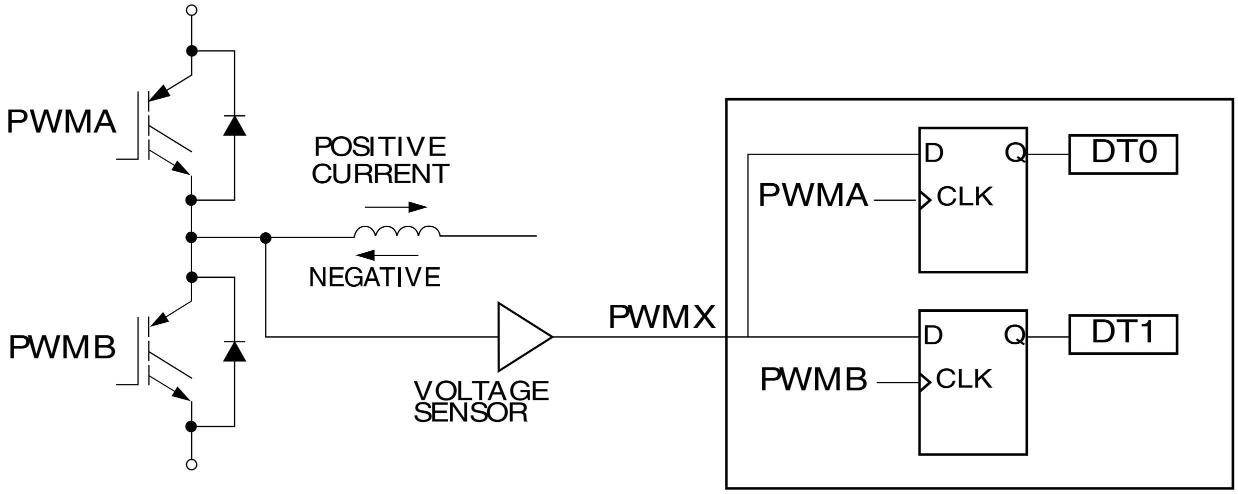


Figure 40-23. Current-status Sense Scheme for Deadtime Correction

Both D flip-flops latch low, DT0 = 0, DT1 = 0, during deadtime periods if current is large and flowing out of the complementary circuit. See the figure above. Both D flip-flops latch the high, DT0 = 1, DT1 = 1, during deadtime periods if current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is somewhere between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. Sampled results will be DT0 = 0 and DT1 = 1. Thus, the best time to change one PWM value register to another is just before the current zero crossing.

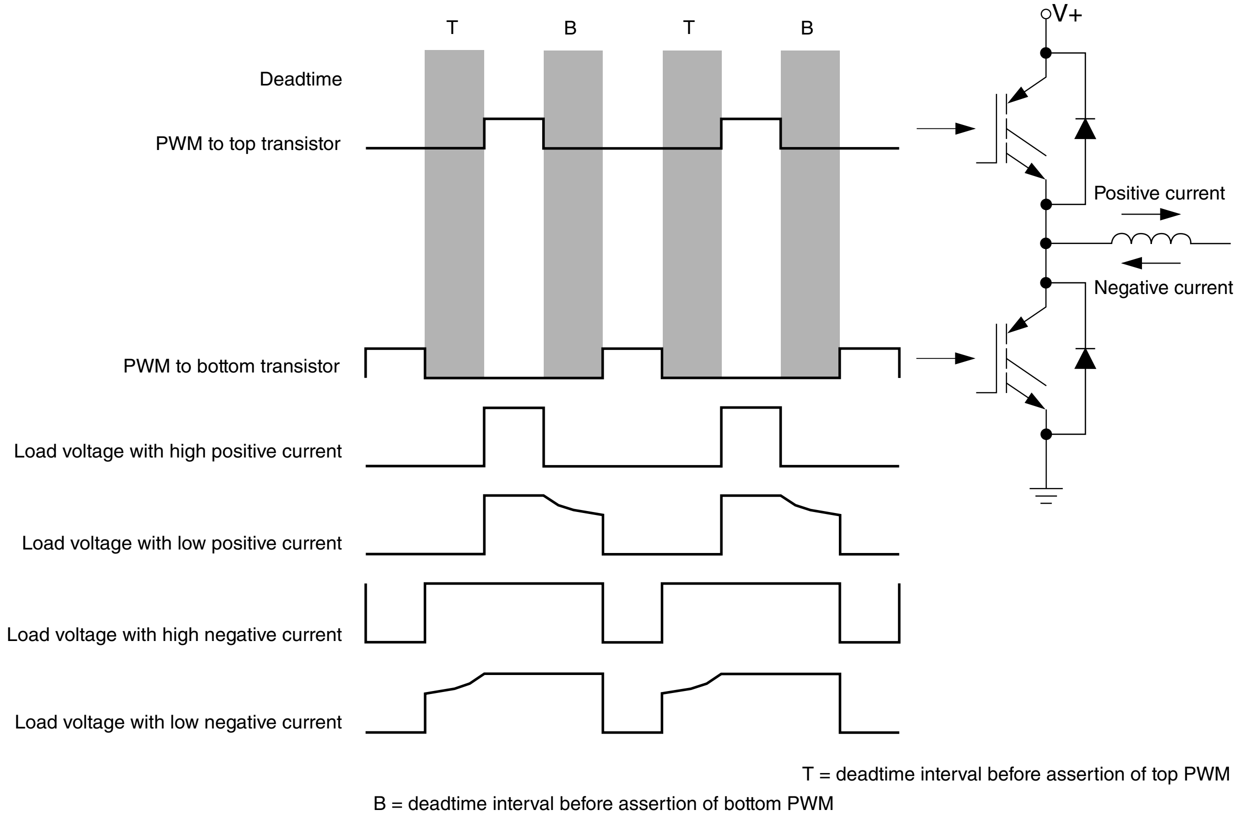

T
B
T
B
V+
Deadtime
PWM to top transistor
Positive current
Negative current
PWM to bottom transistor
Load voltage with high positive current
Load voltage with low positive current
Load voltage with high negative current
Load voltage with low negative current
T = deadtime interval before assertion of top PWM
B = deadtime interval before assertion of bottom PWM

Figure 40-24. Output Voltage Waveforms

40.4.2.9 Output Logic

The following figure shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing to the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic in the figure are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program the

POLA and POLB bits before enabling the output pins. A fault condition can result in the PWM output being tristated, forced to a logic 1, or forced to a logic 0 depending on the values programmed into the PWMxFS fields.

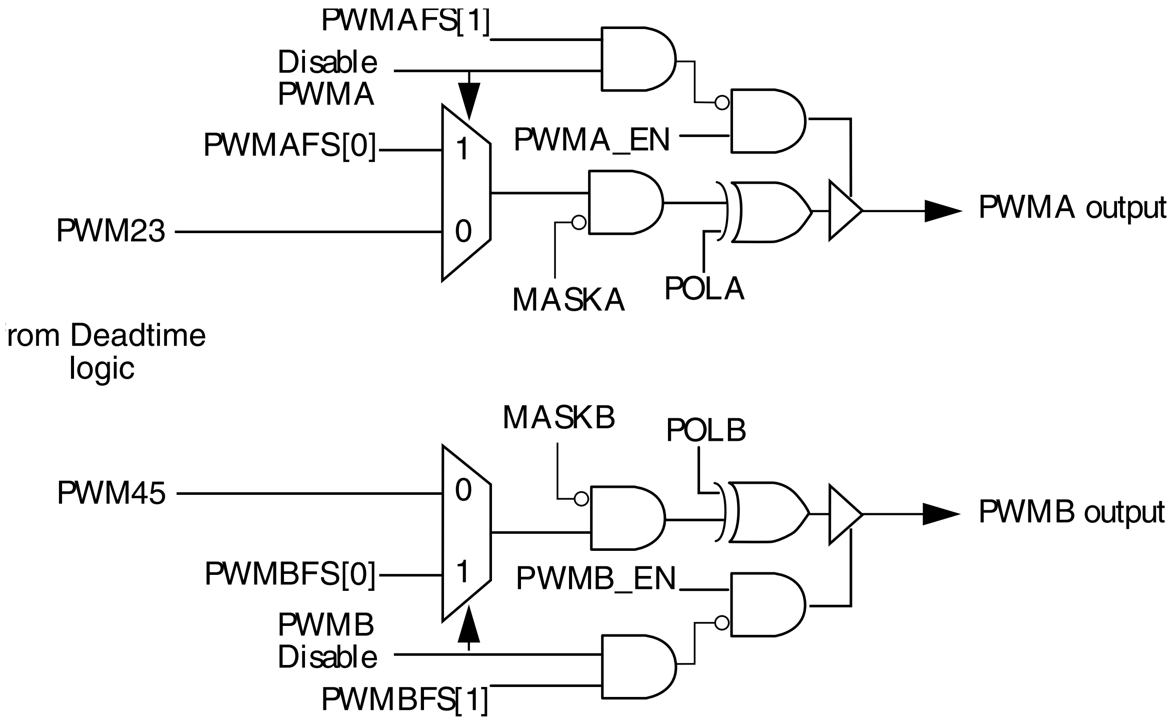


Figure 40-25. Output Logic Section

40.4.2.10 E-Capture

Commensurate with the idea of controlling both edges of an output signal, the Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

The following figure illustrates the block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8-bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8-bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. this feature is useful for dividing down high frequency signals for capture processing so that capture interrupts don't overwhelm the CPU. Also, this feature can be used to simply generate an interrupt after "n" events have been counted.

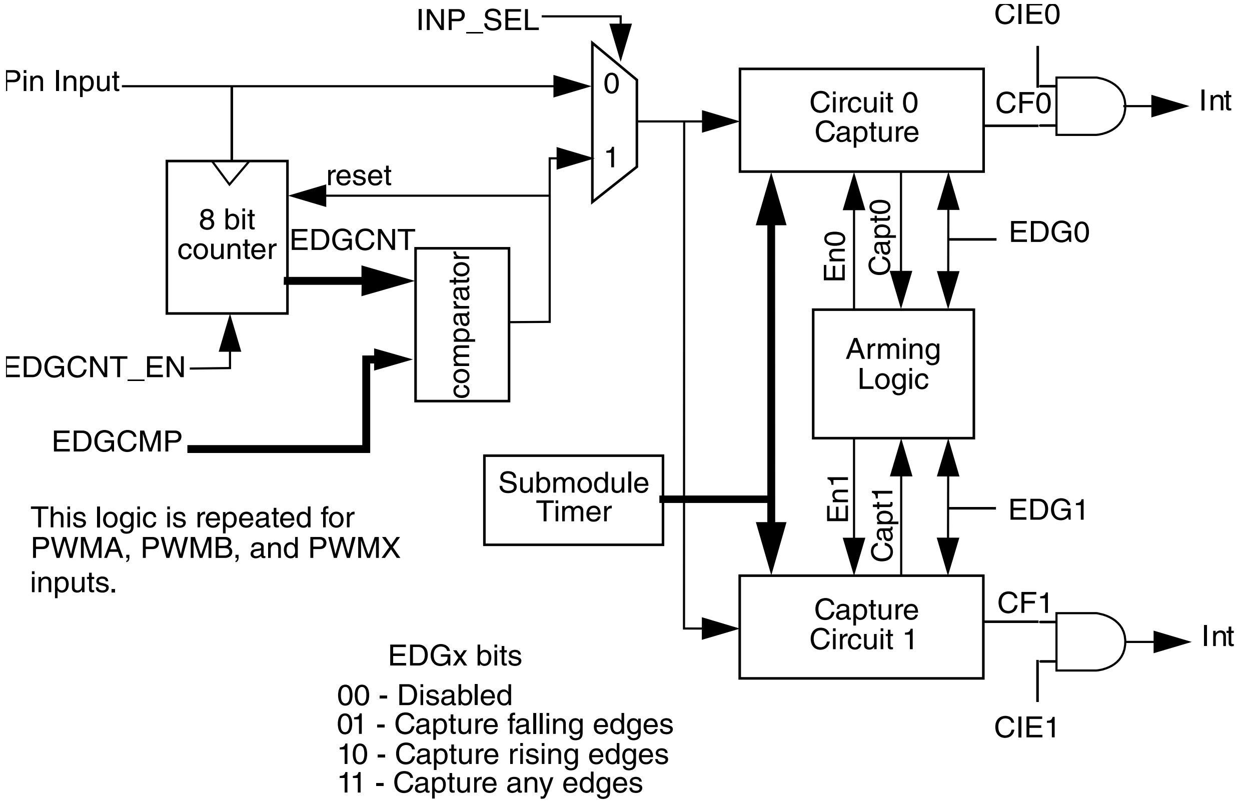


Figure 40-26. Enhanced Capture (E-Capture) Logic

Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by the EDGx1 and EDGx0 bits whose functionality is listed in the figure above. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

40.4.2.11 Fault Protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via the FLVL bits. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or tristated depending the values of the PWMxFS bits.

The fault decoder disables PWM pins selected by the fault logic and the disable mapping register (DISMAP). See the following figure for an example of the fault disable logic. Each bank of bits in DISMAP control the mapping for a single PWM pin. Refer to following table.

The fault protection is enabled even when the PWM module is not enabled; therefore, a fault will be latched in and must be cleared in order to prevent an interrupt when the PWM is enabled.

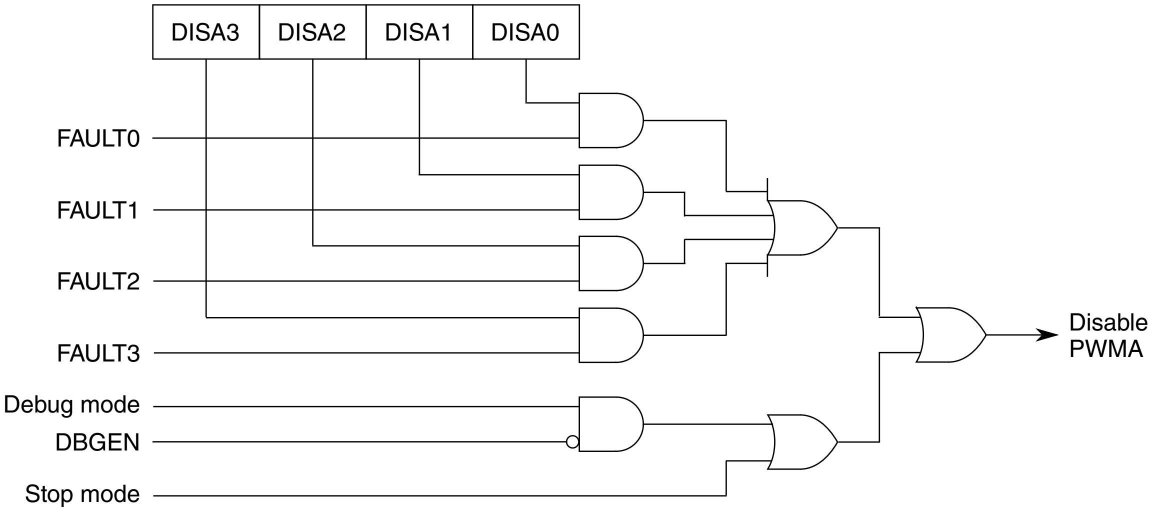


Figure 40-27. Fault decoder for PWMA

Table 40-2. Fault Mapping

PWM Pin	Controlling Register Bits
PWMA	DISA[3:0]
PWMB	DISB[3:0]
PWMX	DISX[3:0]

40.4.2.11.1 Fault Pin Filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with the FILT_PER field of the FFILTx register. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using the FILT_CNT field of the same register. Setting FILT_PER to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FLVLx is set), the corresponding FFPINx and fault flag, FFLAGx, bits are set. The FFPINx bit remains set as long as the filtered FAULTx pin is zero. Clear FFLAGx by writing a logic 1 to FFLAGx.

If the FIEx, FAULTx pin interrupt enable bit is set, the FFLAGx flag generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears the FFLAGx flag by writing a logic one to the bit
- Software clears the FIEx bit by writing a logic zero to it
- A reset occurs

Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

40.4.2.11.2 Automatic Fault Clearing

Setting an automatic clearing mode bit, FAUTOx, configures faults from the FAULTx pin for automatic clearing.

When FAUTOx is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See the following figure. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle. Clearing the FFLAGx flag does not affect disabled PWM pins when FAUTOx is set.

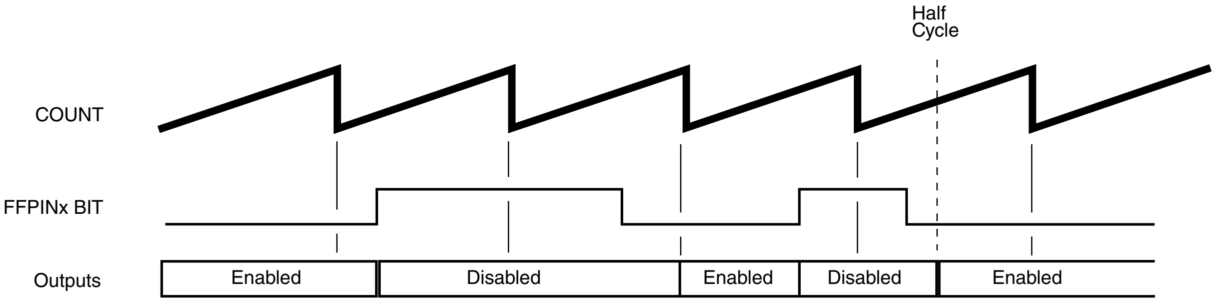


Figure 40-28. Automatic Fault Clearing

40.4.2.11.3 Manual Fault Clearing

Clearing the automatic clearing mode bit, FAUTOx, configures faults from the FAULTx pin for manual clearing:

- If the fail safe mode bits, FSAFEx, are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FFLAGx flag
  - The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin. See the following figure. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle.
- If the fail safe mode bits, FSAFEx, are set, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FFLAGx flag
  - The filter detects a logic one on the FAULTx pin at the start of the next PWM full or half cycle boundary. See Figure 40-30. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle.

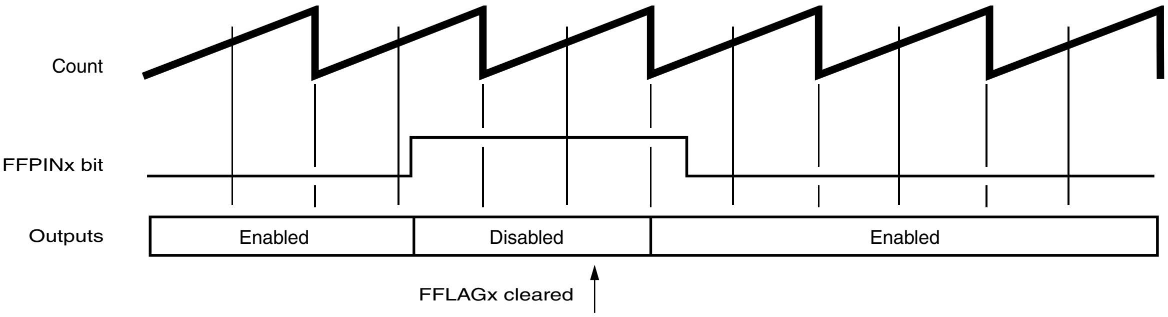


Figure 40-29. Manual Fault Clearing (FSAFE=0)

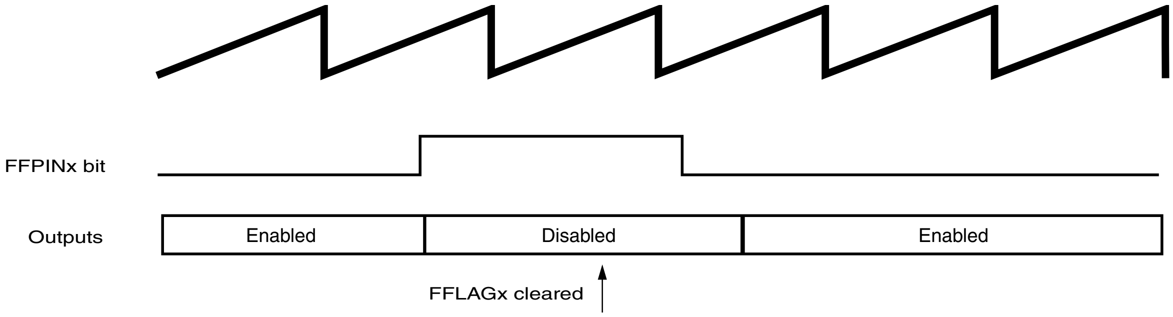


Figure 40-30. Manual Fault Clearing (FSAFE=1)

Note

> Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or EXTA and EXTB. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, RUN equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, RUN equals zero. Thus, fault clearing occurs at peripheral cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

40.4.2.11.4 Fault Testing

The FTEST bit is used to simulate a fault condition on each of the fault inputs.

40.4.3 PWM Generator Loading

This section describes load enabling, load frequency, the reload flag, reload errors, and initialization of the PWM generator.

40.4.3.1 Load Enable

The LDOK bit enables loading of the following PWM generator parameters:

- The prescaler divisor—from SUB*n*_CTRL1[PRSC]
- The PWM period and pulse width—from the SUB*n*_INIT and SUB*n*_VAL*x* registers

LDOK allows software to finish calculating all of these PWM parameters so they can be synchronously updated. SUB*n*_CTRL1[PRSC], SUB*n*_INIT and SUB*n*_VAL*x* are loaded by software into a set of outer buffers. When LDOK is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator. These values can be transfered to the inner set of registers immediately upon setting LDOK if LDMOD is set. Set LDOK by reading it when it is a logic zero and then writing a logic one to it. After loading, LDOK is automatically cleared.

40.4.3.2 Load Frequency

The LDFQ bits in the CTRL1 register select an integral loading frequency of one to 16 PWM reload opportunities. The LDFQ bits take effect at every PWM reload opportunity, regardless the state of the LDOK bit. The HALF and FULL bits in the CTRL1 register control reload timing. If FULL is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If HALF is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both HALF and FULL are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

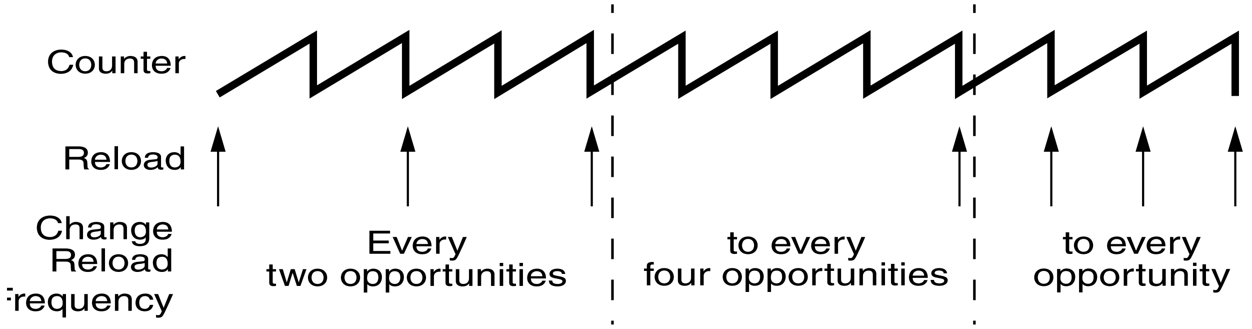


Figure 40-31. Full Cycle Reload Frequency Change

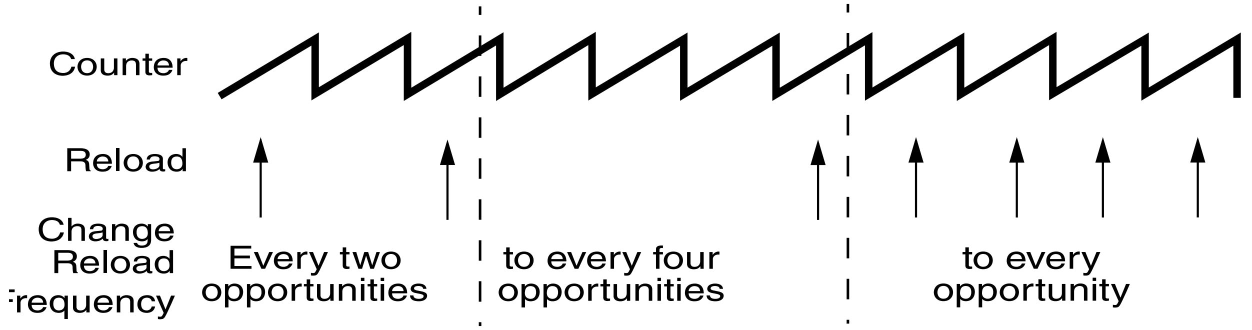


Figure 40-32. Half Cycle Reload Frequency Change

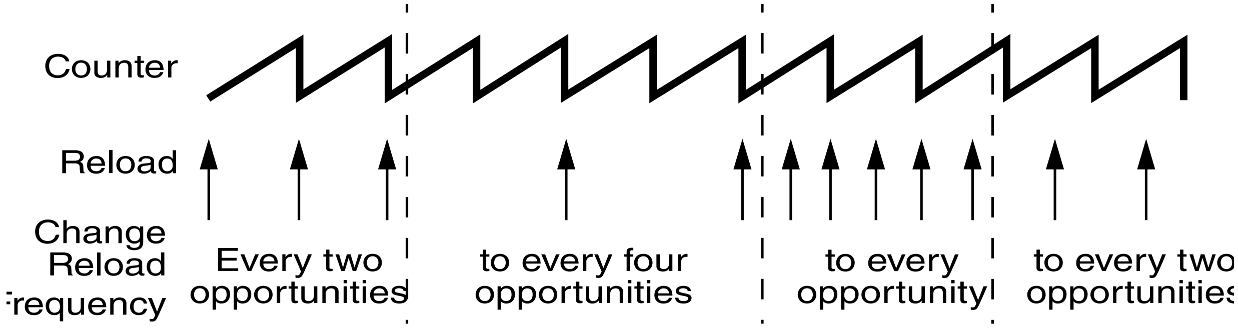


Figure 40-33. Full and Half Cycle Reload Frequency Change

40.4.3.3 Reload Flag

At every reload opportunity the PWM Reload Flag (RF) in the FlexPWM_SUBn_STS register is set. Setting RF happens even if an actual reload is prevented by the LDOK bit. If the PWM reload interrupt enable bit, RIE is set, the RF flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When RIE is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

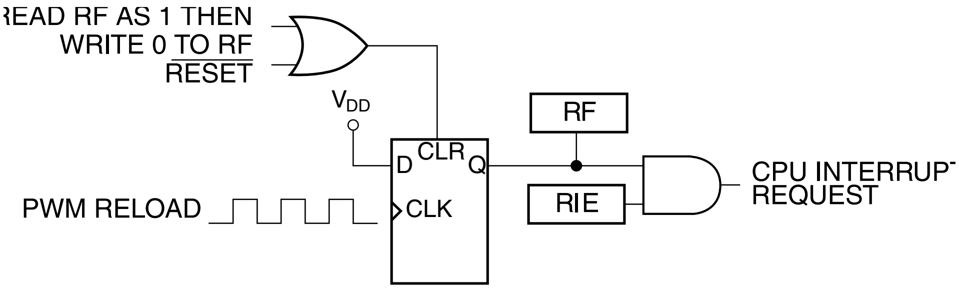


Figure 40-34. PWMF Reload Interrupt Request

40.4.3.4 Reload Errors

Whenever either SUB*n*_VAL*x* or SUB*n*_CTRL1[PSRC] is updated, the RUF flag is set to indicate that the data is not coherent. RUF will be cleared by a successful reload which consists of the reload signal while LDOK is set. If RUF is set and LDOK is clear when the reload signal occurs, a reload error has taken place and the REF bit is set. If RUF is clear when a reload signal asserts, then the data is coherent and no error will be flagged.

40.4.3.5 Initialization

Initialize all registers and set the LDOK bit before setting the RUN bit.

Note

> Even if LDOK is not set, setting RUN also sets the RF flag. To prevent a CPU interrupt request, clear the RIE bit before setting RUN.

The PWM generator uses the last values loaded if RUN is cleared and then set while LDOK equals zero.

When the RUN bit is cleared:

- The RF flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active
- Software/external output control remains active
- Deadtime insertion continues during software/external output control

40.5 Resets

A reset will force all registers to their reset states and tri-state the PWM outputs.

40.6 Interrupts

Each of the submodules within the FlexPWM can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 40-3. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
ipi_int_comp0	CMPF_0	CMPIE_0	Submodule 0 compare interrupt	Compare event has occurred
ipi_int_capt0	CFX1_0, CFX0_0	CFX1IE_0, CFX0IE_0	Submodule 0 input capture interrupt	Input capture event has occurred
ipi_int_reload0	RF_0	RIE_0	Submodule 0 reload interrupt	Reload event has occurred
ipi_int_comp1	CMPF_1	CMPIE_1	Submodule 1 compare interrupt	Compare event has occurred
ipi_int_capt1	CFX1_1, CFX0_1	CFX1IE_1, CFX0IE_1	Submodule 1 input capture interrupt	Input capture event has occurred

Table continues on the next page...

Table 40-3. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
ipi_int_reload1	RF_1	RIE_1	Submodule 1 reload interrupt	Reload event has occurred
ipi_int_comp2	CMPF_2	CMPIE_2	Submodule 2 compare interrupt	Compare event has occurred
ipi_int_capt2	CFX1_2, CFX0_2	CFX1IE_2, CFX0IE_2	Submodule 2 input capture interrupt	Input capture event has occurred
ipi_int_reload2	RF_2	RIE_2	Submodule 2 reload interrupt	Reload event has occurred
ipi_int_comp3	CMPF_3	CMPIE_3	Submodule 3 compare interrupt	Compare event has occurred
ipi_int_capt3	CFX1_3, CFX0_3	CFX1IE_3, CFX0IE_3	Submodule 3 input capture interrupt	Input capture event has occurred
ipi_int_reload3	RF_3	RIE_3	Submodule 3 reload interrupt	Reload event has occurred
ipi_int_rerr	REF_0	REIE_0	Submodule 0 reload error interrupt	Reload error has occurred
	REF_1	REIE_1	Submodule 1 reload error interrupt	
	REF_2	REIE_2	Submodule 2 reload error interrupt	
	REF_3	REIE_3	Submodule 3 reload error interrupt	
ipi_int_fault	FFLAG	FIE	Fault input interrupt	Fault condition has been detected

40.7 DMA

Each submodule can request a DMA read access for its capture FIFOs and a DMA write request for its double buffered SUB*n*_VAL*x* registers.

Table 40-4. DMA Summary

DMA Request	DMA Enable	Name	Description
Submodule 0 read request	CX0DE_0	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_0	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_0	Capture FIFO read request source select	Selects source of read DMA request
Submodule 0 write request	VALDE_0	SUB*n*_VAL*x* write request	SUB*n*_VAL*x* registers need to be updated
Submodule 1 read request	CX0DE_1	Capture FIFO X0 read request	CVAL0 contains a value to be read

Table continues on the next page...

Table 40-4. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
	CX1DE_1	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_1	Capture FIFO read request source select	Selects source of read DMA request
Submodule 1 write request	VALDE_1	SUB*n*_VAL*x* write request	SUB*n*_VAL*x* registers need to be updated
Submodule 2 read request	CX0DE_2	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_2	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_2	Capture FIFO read request source select	Selects source of read DMA request
Submodule 2 write request	VALDE_2	SUB*n*_VAL*x* write request	SUB*n*_VAL*x* registers need to be updated
Submodule 3 read request	CX0DE_3	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_3	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_3	Capture FIFO read request source select	Selects source of read DMA request
Submodule 3 write request	VALDE_3	SUB*n*_VAL*x* write request	SUB*n*_VAL*x* registers need to be updated

Chapter 41
Cross-Triggering Unit (CTU)

41.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

In PWM driven systems it is important to schedule the acquisition of the state variables with respect to the PWM cycle. State variables are obtained through the following peripherals: ADC, position counter (for example, quadrature decoder, resolver, and sine-cos sensor), and PWM duty cycle decoder.

The Cross-Triggering Unit (CTU) is intended to completely avoid CPU involvement in the acquisition of state variables during a control cycle defined by the PWM cycle, the half PWM cycle, or a multiple number of PWM cycles. Several CTU registers are double-buffered, thus allowing coherent update during the transition from one control cycle to the next. Thus double-buffered registers updated at control cycle N will be updated during the transition from control cycle N to control cycle N+1.

The CTU provides four FIFO memory blocks to temporarily store ADC results for CPU access.

41.2 Block diagram

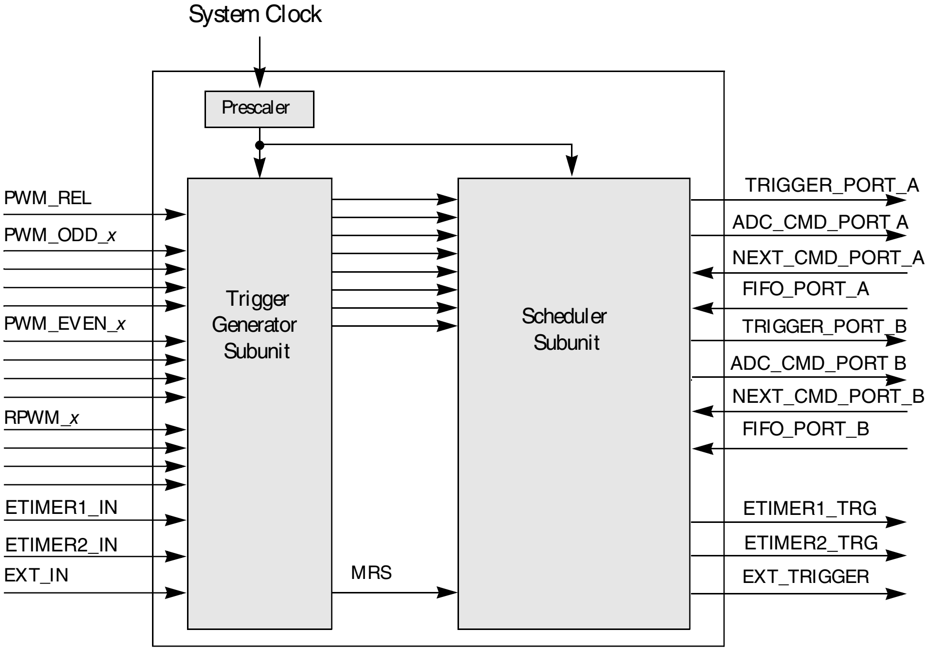


Figure 41-1. CTU block diagram

41.3 CTU overview

The CTU receives several signals from different sources such as PWM, timers, position decoder, and/or external pins. These signals are then processed to generate up to eight trigger events. An input event can be a rising edge, a falling edge, or both edges of the incoming signal. The output can be a pulse, an ADC command, or a stream of consecutive ADC commands for oversampling support. The outputs are targeted to one or more peripherals such as ADCs and timers.

The CTU implements the following features:

- Cross triggering between ADC, PWM timers, on-chip general purpose timers, and external pins
- Double-buffered trigger unit that generates up to 8 triggers
- Trigger generation unit configurable in sequential or triggered modes
- Trigger delay functionality to compensate for the delay of external low pass filters
- Double-buffered ADC command list pointers
- Double-buffered ADC conversion command list
- Each trigger has the capability to generate consecutive conversion commands

- ADC conversion commands allow two ADCs to be controlled synchronously with queue selection to store conversion results
- DMA support

In a typical application, the CTU interfaces with the following peripherals:

- PWM generators for motor control
- Timers
- External signals

The inputs are 16 digital signals. The CTU detects the rising and/or falling edges for each one of them.

Figure 41-1 shows a high level CTU block diagram. The CTU hardware architecture consists of two subunits:

- Trigger generator subunit
- Scheduler subunit

The Trigger generator subunit receives CTU input signals and selects the active edges in order to generate the Master Reload signal. The Master Reload is used to load several CTU double buffered registers. See sections TGS in triggered mode and TGS in sequential mode for more details about the Master Reload signal. The Trigger generator subunit also generates up to eight trigger event signals which are used by the Scheduler subunit.

The Scheduler subunit generates the trigger event output according to the trigger event signals from the Trigger generator subunit. The trigger event output starts an ADC conversion or generates output signals to trigger events for IPs in the SoC. See section Scheduler subunit (SU) for more detail about the Scheduler subunit.

41.3.1 Interaction with other peripherals

The following figure shows an example of how the CTU interacts with on chip peripherals.

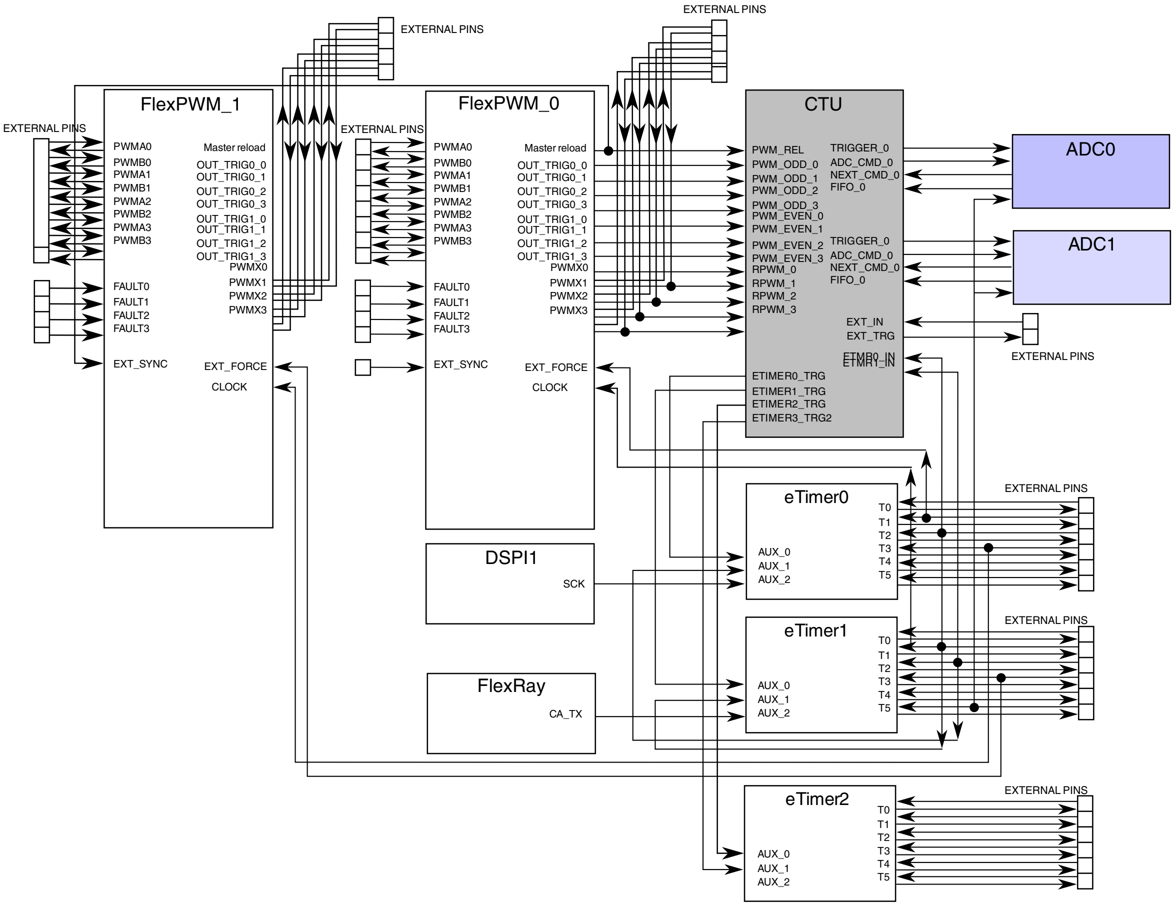


Figure 41-2. CTU application example: interaction with other peripherals

NOTE

For the specific manner in which the CTU interacts with other peripherals on your particular device, see the device configuration details.

41.3.2 Modes of operation

The CTU has two main modes of operation:

1. Triggered mode: the input event from the CTU interface is used to generate a sequence of up to 8 triggers to several outputs such as the ADCs, timers, and external triggers. Internal sequencer logic is used to schedule the triggers based upon the input event occurrence.
2. Sequential mode: each input event generates only one trigger for the selected output, such as ADCs, timers, and external triggers.

41.4 Signal description

The following table shows the chip-level signals for the CTU module.

Table 41-1. Signal properties

Name	Function
EXT_IN	Input pin for external triggers
EXT_TRG	Output pin for external triggers

41.5 Memory Map and Registers

CTU registers are implemented in two sizes:

- 32-bit registers: byte access for write operations and 32-bit access for read operation
- 16-bit registers: byte access for write operations and 32-bit access for read operations

If a 32-bit write operation is performed over a 16-bit register, the write operation is performed on the 4 aligned bytes. Read operation should be performed only at 32-bit boundary.

NOTE

Reserved bit registers, when available for write operations, should be written always with logic 0. Read access does not return meaningful data from reserved bit registers. This module does not issue transfer errors when trying to access its reserved locations.

NOTE

If the system attempts to access an invalid address within the CTU memory map, such as a non-existent register, it does not assert a slave bus error signal to the system. So the user should not rely on such an error when accessing the CTU.

The following tables describe the CTU memory mapped registers.

CTU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Trigger Generator Subunit Input Selection Register (CTU_TGSISR)	32	R/W	0000_0000h	41.5.1/1214
4	Trigger Generator Subunit Control Register (CTU_TGSCR)	16	R/W	0000h	41.5.2/1218
6	Trigger Compare Register (CTU_T0CR)	16	R/W	0000h	41.5.3/1219
8	Trigger Compare Register (CTU_T1CR)	16	R/W	0000h	41.5.3/1219
A	Trigger Compare Register (CTU_T2CR)	16	R/W	0000h	41.5.3/1219
C	Trigger Compare Register (CTU_T3CR)	16	R/W	0000h	41.5.3/1219
E	Trigger Compare Register (CTU_T4CR)	16	R/W	0000h	41.5.3/1219
10	Trigger Compare Register (CTU_T5CR)	16	R/W	0000h	41.5.3/1219
12	Trigger Compare Register (CTU_T6CR)	16	R/W	0000h	41.5.3/1219
14	Trigger Compare Register (CTU_T7CR)	16	R/W	0000h	41.5.3/1219
16	TGS Counter Compare Register (CTU_TGSCCR)	16	R/W	0000h	41.5.4/1219
18	TGS Counter Reload Register (CTU_TGSCRR)	16	R/W	0000h	41.5.5/1219
1C	Commands List Control Register 1 (CTU_CLCR1)	32	R/W	0000_0000h	41.5.6/1220
20	Commands List Control Register 2 (CTU_CLCR2)	32	R/W	0000_0000h	41.5.7/1220
24	Trigger Handler Control Register 1 (CTU_THCR1)	32	R/W	0000_0000h	41.5.8/1221
28	Trigger Handler Control Register 2 (CTU_THCR2)	32	R/W	0000_0000h	41.5.9/1225
2C	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_1)	16	R/W	0000h	41.5.10/ 1228
2C	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_1)	16	R/W	0000h	41.5.11/ 1229
2C	Command List Register C for self-test commands (CTU_CLR_C_1)	16	R/W	0000h	41.5.12/ 1230
2E	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_2)	16	R/W	0000h	41.5.10/ 1228
2E	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_2)	16	R/W	0000h	41.5.11/ 1229
2E	Command List Register C for self-test commands (CTU_CLR_C_2)	16	R/W	0000h	41.5.12/ 1230
30	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_3)	16	R/W	0000h	41.5.10/ 1228
30	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_3)	16	R/W	0000h	41.5.11/ 1229
30	Command List Register C for self-test commands (CTU_CLR_C_3)	16	R/W	0000h	41.5.12/ 1230
32	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_4)	16	R/W	0000h	41.5.10/ 1228
32	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_4)	16	R/W	0000h	41.5.11/ 1229
32	Command List Register C for self-test commands (CTU_CLR_C_4)	16	R/W	0000h	41.5.12/ 1230
34	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_5)	16	R/W	0000h	41.5.10/ 1228

Table continues on the next page...

CTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_5)	16	R/W	0000h	41.5.11/ 1229
34	Command List Register C for self-test commands (CTU_CLR_C_5)	16	R/W	0000h	41.5.12/ 1230
36	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_6)	16	R/W	0000h	41.5.10/ 1228
36	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_6)	16	R/W	0000h	41.5.11/ 1229
36	Command List Register C for self-test commands (CTU_CLR_C_6)	16	R/W	0000h	41.5.12/ 1230
38	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_7)	16	R/W	0000h	41.5.10/ 1228
38	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_7)	16	R/W	0000h	41.5.11/ 1229
38	Command List Register C for self-test commands (CTU_CLR_C_7)	16	R/W	0000h	41.5.12/ 1230
3A	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_8)	16	R/W	0000h	41.5.10/ 1228
3A	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_8)	16	R/W	0000h	41.5.11/ 1229
3A	Command List Register C for self-test commands (CTU_CLR_C_8)	16	R/W	0000h	41.5.12/ 1230
3C	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_9)	16	R/W	0000h	41.5.10/ 1228
3C	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_9)	16	R/W	0000h	41.5.11/ 1229
3C	Command List Register C for self-test commands (CTU_CLR_C_9)	16	R/W	0000h	41.5.12/ 1230
3E	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_10)	16	R/W	0000h	41.5.10/ 1228
3E	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_10)	16	R/W	0000h	41.5.11/ 1229
3E	Command List Register C for self-test commands (CTU_CLR_C_10)	16	R/W	0000h	41.5.12/ 1230
40	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_11)	16	R/W	0000h	41.5.10/ 1228
40	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_11)	16	R/W	0000h	41.5.11/ 1229
40	Command List Register C for self-test commands (CTU_CLR_C_11)	16	R/W	0000h	41.5.12/ 1230
42	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_12)	16	R/W	0000h	41.5.10/ 1228
42	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_12)	16	R/W	0000h	41.5.11/ 1229

Table continues on the next page...

CTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
42	Command List Register C for self-test commands (CTU_CLR_C_12)	16	R/W	0000h	41.5.12/ 1230
44	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_13)	16	R/W	0000h	41.5.10/ 1228
44	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_13)	16	R/W	0000h	41.5.11/ 1229
44	Command List Register C for self-test commands (CTU_CLR_C_13)	16	R/W	0000h	41.5.12/ 1230
46	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_14)	16	R/W	0000h	41.5.10/ 1228
46	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_14)	16	R/W	0000h	41.5.11/ 1229
46	Command List Register C for self-test commands (CTU_CLR_C_14)	16	R/W	0000h	41.5.12/ 1230
48	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_15)	16	R/W	0000h	41.5.10/ 1228
48	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_15)	16	R/W	0000h	41.5.11/ 1229
48	Command List Register C for self-test commands (CTU_CLR_C_15)	16	R/W	0000h	41.5.12/ 1230
4A	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_16)	16	R/W	0000h	41.5.10/ 1228
4A	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_16)	16	R/W	0000h	41.5.11/ 1229
4A	Command List Register C for self-test commands (CTU_CLR_C_16)	16	R/W	0000h	41.5.12/ 1230
4C	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_17)	16	R/W	0000h	41.5.10/ 1228
4C	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_17)	16	R/W	0000h	41.5.11/ 1229
4C	Command List Register C for self-test commands (CTU_CLR_C_17)	16	R/W	0000h	41.5.12/ 1230
4E	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_18)	16	R/W	0000h	41.5.10/ 1228
4E	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_18)	16	R/W	0000h	41.5.11/ 1229
4E	Command List Register C for self-test commands (CTU_CLR_C_18)	16	R/W	0000h	41.5.12/ 1230
50	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_19)	16	R/W	0000h	41.5.10/ 1228
50	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_19)	16	R/W	0000h	41.5.11/ 1229
50	Command List Register C for self-test commands (CTU_CLR_C_19)	16	R/W	0000h	41.5.12/ 1230

Table continues on the next page...

CTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
52	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_20)	16	R/W	0000h	41.5.10/ 1228
52	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_20)	16	R/W	0000h	41.5.11/ 1229
52	Command List Register C for self-test commands (CTU_CLR_C_20)	16	R/W	0000h	41.5.12/ 1230
54	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_21)	16	R/W	0000h	41.5.10/ 1228
54	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_21)	16	R/W	0000h	41.5.11/ 1229
54	Command List Register C for self-test commands (CTU_CLR_C_21)	16	R/W	0000h	41.5.12/ 1230
56	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_22)	16	R/W	0000h	41.5.10/ 1228
56	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_22)	16	R/W	0000h	41.5.11/ 1229
56	Command List Register C for self-test commands (CTU_CLR_C_22)	16	R/W	0000h	41.5.12/ 1230
58	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_23)	16	R/W	0000h	41.5.10/ 1228
58	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_23)	16	R/W	0000h	41.5.11/ 1229
58	Command List Register C for self-test commands (CTU_CLR_C_23)	16	R/W	0000h	41.5.12/ 1230
5A	Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_24)	16	R/W	0000h	41.5.10/ 1228
5A	Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_24)	16	R/W	0000h	41.5.11/ 1229
5A	Command List Register C for self-test commands (CTU_CLR_C_24)	16	R/W	0000h	41.5.12/ 1230
6C	FIFO DMA Control Register (CTU_FDCR)	16	R/W	0F00h	41.5.13/ 1231
70	FIFO Control Register (CTU_FCR)	32	R/W	0000_0000h	41.5.14/ 1233
74	FIFO Threshold Register (CTU_FTH)	32	R/W	0000_0000h	41.5.15/ 1235
7C	FIFO Status Register (CTU_FST)	32	R	0000_2222h	41.5.16/ 1236
80	FIFO Right Aligned Data Register (CTU_FR0)	32	R	0000_0000h	41.5.17/ 1238
84	FIFO Right Aligned Data Register (CTU_FR1)	32	R	0000_0000h	41.5.17/ 1238
88	FIFO Right Aligned Data Register (CTU_FR2)	32	R	0000_0000h	41.5.17/ 1238

Table continues on the next page...

CTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8C	FIFO Right Aligned Data Register (CTU_FR3)	32	R	0000_0000h	41.5.17/ 1238
A0	FIFO Signed Left Aligned Data Register (CTU_FL0)	32	R	0000_0000h	41.5.18/ 1239
A4	FIFO Signed Left Aligned Data Register (CTU_FL1)	32	R	0000_0000h	41.5.18/ 1239
A8	FIFO Signed Left Aligned Data Register (CTU_FL2)	32	R	0000_0000h	41.5.18/ 1239
AC	FIFO Signed Left Aligned Data Register (CTU_FL3)	32	R	0000_0000h	41.5.18/ 1239
C0	Error Flag Register (CTU_EFR)	16	w1c	0000h	41.5.19/ 1240
C2	Interrupt Flag Register (CTU_IFR)	16	w1c	0000h	41.5.20/ 1242
C4	Interrupt/DMA Register (CTU_IR)	16	R/W	0000h	41.5.21/ 1243
C6	Control ON Time Register (CTU_COTR)	16	R/W	0000h	41.5.22/ 1245
C8	Control Register (CTU_CR)	16	R/W	0000h	41.5.23/ 1245
CA	Digital Filter Register (CTU_DFR)	16	R/W	0000h	41.5.24/ 1247
CC	Expected Value A Register (CTU_EXPAR)	16	R/W	FFFFh	41.5.25/ 1248
CE	Expected Value B Register (CTU_EXPBR)	16	R/W	FFFFh	41.5.26/ 1248
D0	Counter Range Register (CTU_CNTRNGR)	16	R/W	0000h	41.5.27/ 1249
D4	List Control/Status Register (CTU_LISTCSR)	32	R/W	0000_0000h	41.5.28/ 1250

41.5.1 Trigger Generator Subunit Input Selection Register (CTU_TGSISR)

The TGSISR register is double buffered. The load from the corresponding buffer to the register is controlled by the TGSISR_RE bit in the CTU_CR register. The TGSISR_RE bit is self negated. Once set, this bit remains set until the TGSISR register is updated from the corresponding buffer register. Reads to the TGSISR register actually return the content of the buffer and not the content of the register. Thus, a read will return the most recent data written to the register address even though the buffer data is not synchronized with the register data.

NOTE

For the particular input connection assigned to each field in this register, see the CTU's chip-specific configuration details.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	I15_FE	I15_RE	I14_FE	I14_RE	I13_FE	I13_RE	I12_FE	I12_RE	I11_FE	I11_RE	I10_FE	I10_RE	I9_FE	I9_RE	I8_FE	I8_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	I7_FE	I7_RE	I6_FE	I6_RE	I5_FE	I5_RE	I4_FE	I4_RE	I3_FE	I3_RE	I2_FE	I2_RE	I1_FE	I1_RE	I0_FE	I0_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_TGSISR field descriptions

Field	Description
0 I15_FE	Input 15 Falling Edge Enable. 0 Disabled 1 Enabled
1 I15_RE	Input 15 Rising Edge Enable. 0 Disabled 1 Enabled
2 I14_FE	Input 14 Falling Edge Enable. 0 Disabled 1 Enabled
3 I14_RE	Input 14 Rising Edge Enable. 0 Disabled 1 Enabled
4 I13_FE	Input 13 Falling Edge Enable. 0 Disabled 1 Enabled
5 I13_RE	Input 13 Rising Edge Enable. 0 Disabled 1 Enabled
6 I12_FE	Input 12 Falling Edge Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_TGSISR field descriptions (continued)

Field	Description
7 I12_RE	Input 12 Rising Edge Enable. 0 Disabled 1 Enabled
8 I11_FE	Input 11 Falling Edge Enable. 0 Disabled 1 Enabled
9 I11_RE	Input 11 Rising Edge Enable. 0 Disabled 1 Enabled
10 I10_FE	Input 10 Falling Edge Enable. 0 Disabled 1 Enabled
11 I10_RE	Input 10 Rising Edge Enable. 0 Disabled 1 Enabled
12 I9_FE	Input 9 Falling Edge Enable. 0 Disabled 1 Enabled
13 I9_RE	Input 9 Rising Edge Enable. 0 Disabled 1 Enabled
14 I8_FE	Input 8 Falling Edge Enable. 0 Disabled 1 Enabled
15 I8_RE	Input 8 Rising Edge Enable. 0 Disabled 1 Enabled
16 I7_FE	Input 7 Falling Edge Enable. 0 Disabled 1 Enabled
17 I7_RE	Input 7 Rising Edge Enable. 0 Disabled 1 Enabled
18 I6_FE	Input 6 Falling Edge Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_TGSISR field descriptions (continued)

Field	Description
19 I6_RE	Input 6 Rising Edge Enable. 0 Disabled 1 Enabled
20 I5_FE	Input 5 Falling Edge Enable. 0 Disabled 1 Enabled
21 I5_RE	Input 5 Rising Edge Enable. 0 Disabled 1 Enabled
22 I4_FE	Input 4 Falling Edge Enable. 0 Disabled 1 Enabled
23 I4_RE	Input 4 Rising Edge Enable. 0 Disabled 1 Enabled
24 I3_FE	Input 3 Falling Edge Enable. 0 Disabled 1 Enabled
25 I3_RE	Input 3 Rising edge Enable 0 Disabled 1 Enabled
26 I2_FE	Input 2 Falling Edge Enable. 0 Disabled 1 Enabled
27 I2_RE	Input 2 Rising Edge Enable. 0 Disabled 1 Enabled
28 I1_FE	Input 1 Falling Edge Enable. 0 Disabled 1 Enabled
29 I1_RE	Input 1 Rising edge Enable 0 Disabled 1 Enabled
30 I0_FE	Input 0 Falling Edge Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_TGSISR field descriptions (continued)

Field	Description
31 I0_RE	Input 0 Rising Edge Enable. 0 Disabled 1 Enabled

41.5.2 Trigger Generator Subunit Control Register (CTU_TGSCR)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7
Read	0							ET_TM
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	PRES		MRS_SM					TGS_M
Reset	0	0	0	0	0	0	0	0

CTU_TGSCR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ET_TM	Enable Toggle Mode for external Trigger. 0 Pulse mode 1 Toggle mode
8–9 PRES	Prescaler selection bits for TGS and SU. This prescaler is used by the TGS counter. 00 Value is 1 01 Value is 2 10 Value is 3 11 Value is 4
10–14 MRS_SM	MRS Selection in Sequential Mode (5 bits to select one of the 32 inputs shown in the Trigger generator subunit input selection (TGSISR) register.
15 TGS_M	Trigger Generator Subunit Mode. 0 Triggered Mode 1 Sequential Mode

41.5.3 Trigger Compare Register (CTU_T*n*CR)

Address: 0h base + 6h offset + (2d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	TCRV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_T*n*CR field descriptions

Field	Description
0–15 TCRV	Trigger Compare Register Value.

41.5.4 TGS Counter Compare Register (CTU_TGSCCR)

Address: 0h base + 16h offset = 16h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	TGSCCV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_TGSCCR field descriptions

Field	Description
0–15 TGSCCV	TGS Counter Compare Value. The value in this register is compared against the TGS counter. When a match occurs, the TGS counter is stopped.

41.5.5 TGS Counter Reload Register (CTU_TGSCRR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	TGSCRV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_TGSCRR field descriptions

Field	Description
0–15 TGSCRV	TGS Counter Reload Value. When an MRS occurs, the value in this register is used to reload the TGS counter.

41.5.6 Commands List Control Register 1 (CTU_CLCR1)

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			T3_INDEX					0			T2_INDEX					0			T1_INDEX					0			T0_INDEX				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CLCR1 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 T3_INDEX	Trigger 3 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 T2_INDEX	Trigger 2 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 T1_INDEX	Trigger 1 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 T0_INDEX	Trigger 0 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.

41.5.7 Commands List Control Register 2 (CTU_CLCR2)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			T7_INDEX					0			T6_INDEX					0			T5_INDEX					0			T4_INDEX				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CLCR2 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 T7_INDEX	Trigger 7 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CTU_CLCR2 field descriptions (continued)

Field	Description
11–15 T6_INDEX	Trigger 6 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 T5_INDEX	Trigger 5 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 T4_INDEX	Trigger 4 Commands List 1st command address. This is the address of the first command in the LIST associated with this trigger. Valid values range from 0 through 23.

41.5.8 Trigger Handler Control Register 1 (CTU_THCR1)

The THCR1 register is double buffered. The update from the corresponding buffer register is controlled by the MRS signal. The THCR1 and THCR2 registers are together organized into 8 groups of 7 enable bits. Each group is associated with one trigger from the Trigger subunit. Each group of enable bits has 7 enables corresponding to a master trigger enable, External Trigger output enable, 4 eTimer outputs, and the ADC command list enable, respectively. If the master trigger enable Tn_E is cleared, the trigger is disabled and the other trigger enable bits in the group have no effect.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	T3_E	T3_ETE	T3_T4E	T3_T3E	T3_T2E	T3_T1E	T3_ADCE	0	T2_E	T2_ETE	T2_T4E	T2_T3E	T2_T2E	T2_T1E	T2_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	T1_E	T1_ETE	T1_T4E	T1_T3E	T1_T2E	T1_T1E	T1_ADCE	0	T0_E	T0_ETE	T0_T4E	T0_T3E	T0_T2E	T0_T1E	T0_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_THCR1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 T3_E	Trigger 3 Enable. 0 Disabled 1 Enabled
2 T3_ETE	Trigger 3 External Trigger output Enable. 0 Disabled 1 Enabled
3 T3_T4E	Trigger 3 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
4 T3_T3E	Trigger 3 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
5 T3_T2E	Trigger 3 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
6 T3_T1E	Trigger 3 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
7 T3_ADCE	Trigger 3 ADC command output Enable. 0 Disabled 1 Enabled
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 T2_E	Trigger 2 Enable. 0 Disabled 1 Enabled
10 T2_ETE	Trigger 2 External Trigger output Enable. 0 Disabled 1 Enabled
11 T2_T4E	Trigger 2 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
12 T2_T3E	Trigger 2 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_THCR1 field descriptions (continued)

Field	Description
13 T2_T2E	Trigger 2 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
14 T2_T1E	Trigger 2 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
15 T2_ADCE	Trigger 2 ADC command output Enable. 0 Disabled 1 Enabled
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 T1_E	Trigger 1 Enable. 0 Disabled 1 Enabled
18 T1_ETE	Trigger 1 External Trigger output Enable. 0 Disabled 1 Enabled
19 T1_T4E	Trigger 1 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
20 T1_T3E	Trigger 1 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
21 T1_T2E	Trigger 1 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
22 T1_T1E	Trigger 1 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
23 T1_ADCE	Trigger 1 ADC command output Enable. 0 Disabled 1 Enabled
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 T0_E	Trigger 0 Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_THCR1 field descriptions (continued)

Field	Description
26 T0_ETE	Trigger 0 External Trigger output Enable. 0 Disabled 1 Enabled
27 T0_T4E	Trigger 0 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
28 T0_T3E	Trigger 0 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
29 T0_T2E	Trigger 0 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
30 T0_T1E	Trigger 0 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
31 T0_ADCE	Trigger 0 ADC command output Enable. 0 Disabled 1 Enabled

41.5.9 Trigger Handler Control Register 2 (CTU_THCR2)

The THCR2 register is double buffered. The update from the corresponding buffer register is controlled by the MRS signal. The THCR1 and THCR2 registers are together organized into 8 groups of 7 enable bits. Each group is associated with one trigger from the Trigger subunit. Each group of enable bits has 7 enables corresponding to a master trigger enable, External Trigger output enable, 4 eTimer outputs, and the ADC command list enable, respectively. If the master trigger enable Tn_E is cleared, the trigger is disabled and the other trigger enable bits in the group have no effect.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	T7_E	T7_ETE	T7_T4E	T7_T3E	T7_T2E	T7_T1E	T7_ADCE	0	T6_E	T6_ETE	T6_T4E	T6_T3E	T6_T2E	T6_T1E	T6_ADCE
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	T5_E	T5_ETE	T5_T4E	T5_T3E	T5_T2E	T5_T1E	T5_ADCE	0	T4_E	T4_ETE	T4_T4E	T4_T3E	T4_T2E	T4_T1E	T4_ADCE
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_THCR2 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 T7_E	Trigger 7 Enable. 0 Disabled 1 Enabled
2 T7_ETE	Trigger 7 External Trigger output Enable. 0 Disabled 1 Enabled
3 T7_T4E	Trigger 7 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
4 T7_T3E	Trigger 7 ETIMER3_TRG output Enable.

Table continues on the next page...

CTU_THCR2 field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
5 T7_T2E	Trigger 7 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
6 T7_T1E	Trigger 7 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
7 T7_ADCE	Trigger 7 ADC command output Enable. 0 Disabled 1 Enabled
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 T6_E	Trigger 6 Enable. 0 Disabled 1 Enabled
10 T6_ETE	Trigger 6 External Trigger output Enable. 0 Disabled 1 Enabled
11 T6_T4E	Trigger 6 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
12 T6_T3E	Trigger 6 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
13 T6_T2E	Trigger 6 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
14 T6_T1E	Trigger 6 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
15 T6_ADCE	Trigger 6 ADC command output Enable. 0 Disabled 1 Enabled
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 T5_E	Trigger 5 Enable.

Table continues on the next page...

CTU_THCR2 field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
18 T5_ETE	Trigger 5 External Trigger output Enable. 0 Disabled 1 Enabled
19 T5_T4E	Trigger 5 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
20 T5_T3E	Trigger 5 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
21 T5_T2E	Trigger 5 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled
22 T5_T1E	Trigger 5 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
23 T5_ADCE	Trigger 5 ADC command output Enable. 0 Disabled 1 Enabled
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 T4_E	Trigger 4 Enable. 0 Disabled 1 Enabled
26 T4_ETE	Trigger 4 External Trigger output Enable. 0 Disabled 1 Enabled
27 T4_T4E	Trigger 4 ETIMER4_TRG output Enable. 0 Disabled 1 Enabled
28 T4_T3E	Trigger 4 ETIMER3_TRG output Enable. 0 Disabled 1 Enabled
29 T4_T2E	Trigger 4 ETIMER2_TRG output Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_THCR2 field descriptions (continued)

Field	Description
30 T4_T1E	Trigger 4 ETIMER1_TRG output Enable. 0 Disabled 1 Enabled
31 T4_ADCE	Trigger 4 ADC command output Enable. 0 Disabled 1 Enabled

41.5.10 Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_*n*)

The Command List Register (CLR_x_n), where n = 1,.......,24, has one of three formats (CLR_A_n, CLR_B_n, CLR_C_n) depending on the setting of its CMS and ST0 bits. If CMS=0 and ST0=0, the single conversion mode is selected and the CLR register assumes the CLR_A_n format described in the following figure.

Address: 0h base + 2Ch offset + (2d × i), where i=0d to 23d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	CIR	LC	CMS	FIFO			ST0	0			SU	0	CH			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CLR_A_*n* field descriptions

Field	Description
0 CIR	Command execution Interrupt Request enable bit. 0 Disabled 1 Enabled
1 LC	Last Command bit. 0 Not last 1 Last
2 CMS	Conversion Mode Selection. Must be 0 in this register format. 0 Single conversion mode 1 Dual conversion mode
3–5 FIFO	FIFO used for ADC Port A / Port B. See CLR_A[SU] bit for Port A versus Port B selection. 000 FIFO_0 select ---- ------------------- 111 FIFO_7 select

Table continues on the next page...

CTU_CLR_A_*n* field descriptions (continued)

Field	Description
6 ST0	Self Test mode control 0. Must be 0 in this register format.
7–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SU	ADC Port A / Port B selection. 0 ADC Port A selected 1 ADC Port B selected
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CH	ADC Port channel number.

41.5.11 Command List Register B for ADC dual-conversion mode commands (CTU_CLR_B_*n*)

The Command List Register (CLR_x_n), where n = 1,.......,24, has one of three formats (CLR_A_n, CLR_B_n, CLR_C_n) depending on the setting of its CMS and ST0 bits. If CMS=1 and ST0=0, the dual conversion mode is selected and the CLR register assumes the CLR_B_n format described in the following figure.

Address: 0h base + 2Ch offset + (2d × i), where i=0d to 23d

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>Read</td><td rowspan="2">CIR</td><td rowspan="2">LC</td><td rowspan="2">CMS</td><td rowspan="2" colspan="3">FIFO</td><td>ST0</td><td rowspan="2" colspan="4">CH_B</td><td>0</td><td rowspan="2" colspan="4">CH_A</td></tr>
<tr><td>Write</td><td>▨</td><td>▨</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

CTU_CLR_B_*n* field descriptions

Field	Description
0 CIR	Command execution Interrupt Request enable bit. 0 Disabled 1 Enabled
1 LC	Last Command bit. 0 Not last 1 Last
2 CMS	Conversion Mode Selection. Must be 1 in this register format. 0 Single conversion mode 1 Dual conversion mode

Table continues on the next page...

CTU_CLR_B_*n* field descriptions (continued)

Field	Description
3–5 FIFO	FIFO used for ADC Port A / Port B. See CLR_A[SU] bit for Port A versus Port B selection. 000 FIFO_0 select --- ------------------- 111 FIFO_7 select
6 ST0	Sel-Test mode control 0. Must be 0 in this register format.
7–10 CH_B	ADC Port B channel number.
11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CH_A	ADC Port A channel number.

41.5.12 Command List Register C for self-test commands (CTU_CLR_C_*n*)

The Command List Register (CLR_x_n), where n = 1,.......,24, has one of three formats (CLR_A_n, CLR_B_n, CLR_C_n) depending on the setting of its ST0 bit. You must first set ST0 to 1 to set the CLR register in the self-test mode register format. In this case, the CLR register assumes the CLR_C_n format described in the following figure.

Address: 0h base + 2Ch offset + (2d × i), where i=0d to 23d

Bit	0	1	2	3	4	5	6	7
Read Write	CIR	LC	ST1	ST_CMS	ST_SU	0	ST0	ALG
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	ALG	BSIZE						
Reset	0	0	0	0	0	0	0	0

CTU_CLR_C_*n* field descriptions

Field	Description
0 CIR	Command Execution Interrupt Request enable bit (this bit has no influence in this mode). **NOTE:** CIR bit value has no influence in self-test mode. The interrupt request is disabled and cannot be enabled by using this bit in this mode.

Table continues on the next page...

CTU_CLR_C_*n* field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
1 LC	Last command bit. 0 Not last 1 Last
2 ST1	Self Test command control 1. ST1 must be 0 for a self test command.
3 ST_CMS	Conversion Mode Selection for Self Test command. 0 Single Conversion Mode 1 Dual Conversion Mode
4 ST_SU	Self Test Selection Unit bit. Usable only if ST_CMS is 0. 0 ADC Port A 1 ADC Port B
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ST0	Self Test command control 0. ST0 must 1 in this register format.
7–8 ALG	Algorithm scheduled: 00 Algorithm S 01 Reserved 10 Algorithm C 11 Full algorithm (S + C)
9–15 BSIZE	Burst size of the algorithm iteration. 0000000 Single step execution .... Two step execution through 511 step execution 1111111 512 step execution with one trigger

41.5.13 FIFO DMA Control Register (CTU_FDCR)

The FDCR register enables the DMA requests issued by the CTU FIFOs. The DMA request occurs when it is enabled and the corresponding FIFO had reached a threshold defined in register FTH.

NOTE

Interrupt and DMA are controlled independently, thus FIFO DMA operation and interrupts can be enabled at the same time.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7
Read	Reserved				Reserved	Reserved	Reserved	Reserved
Write	w1c				w1c	w1c	w1c	w1c
Reset	0	0	0	0	1	1	1	1

Bit	8	9	10	11	12	13	14	15
Read	Reserved				DE3	DE2	DE1	DE0
Write	w1c							
Reset	0	0	0	0	0	0	0	0

CTU_FDCR field descriptions

Field	Description
0–3 Reserved	This field is reserved.
4 Reserved	This field is reserved. **NOTE:** The reset value of this bit is 1b. If this reset value is to be preserved during a write to this bit, 0b must be written. This bit is a "write 1 to clear" bit, which means a write of 1b changes the bit's value to 0b.
5 Reserved	This field is reserved. **NOTE:** The reset value of this bit is 1b. If this reset value is to be preserved during a write to this bit, 0b must be written. This bit is a "write 1 to clear" bit, which means a write of 1b changes the bit's value to 0b.
6 Reserved	This field is reserved. **NOTE:** The reset value of this bit is 1b. If this reset value is to be preserved during a write to this bit, 0b must be written. This bit is a "write 1 to clear" bit, which means a write of 1b changes the bit's value to 0b.
7 Reserved	This field is reserved. **NOTE:** The reset value of this bit is 1b. If this reset value is to be preserved during a write to this bit, 0b must be written. This bit is a "write 1 to clear" bit, which means a write of 1b changes the bit's value to 0b.
8–11 Reserved	This field is reserved. **NOTE:** The reset value of each individual bit is 0b1. If this reset value is to be preserved during a write to these bits, and 0b0 must be written to each of these bits.
12 DE3	FIFO 3 DMA enable. 0 Disabled 1 Enabled
13 DE2	FIFO 2 DMA enable. 0 Disabled 1 Enabled
14 DE1	FIFO 1 DMA enable.

Table continues on the next page...

CTU_FDCR field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
15 DE0	FIFO 0 DMA enable. 0 Disabled 1 Enabled

41.5.14 FIFO Control Register (CTU_FCR)

This register implements interrupt enable bits to signal FIFO behavior. FIFO overrun, overflow, empty, and full indication from one of the FIFOs are OR-ed together to indicate through one interrupt line that one or more flags are active. After receiving an interrupt the FST register should be checked to evaluate what caused that interrupt. There is one interrupt line per FIFO module.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OR_ EN3	OF_ EN3	EMPTY_EN3	FULL_EN3	OR_ EN2	OF_ EN2	EMPTY_EN2	FULL_EN2	OR_ EN1	OF_ EN1	EMPTY_EN1	FULL_EN1	OR_ EN0	OF_ EN0	EMPTY_EN0	FULL_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_FCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 OR_EN3	FIFO 3 Overrun interrupt enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_FCR field descriptions (continued)

Field	Description
17 OF_EN3	FIFO 3 threshold Overflow interrupt enable. 0 Disabled 1 Enabled
18 EMPTY_EN3	FIFO 3 Empty interrupt enable. 0 Disabled 1 Enabled
19 FULL_EN3	FIFO 3 Full interrupt enable. 0 Disabled 1 Enabled
20 OR_EN2	FIFO 2 Overrun interrupt enable. 0 Disabled 1 Enabled
21 OF_EN2	FIFO 2 threshold Overflow interrupt enable. 0 Disabled 1 Enabled
22 EMPTY_EN2	FIFO 2 Empty interrupt enable. 0 Disabled 1 Enabled
23 FULL_EN2	FIFO 2 Full interrupt enable. 0 Disabled 1 Enabled
24 OR_EN1	FIFO 1 Overrun interrupt enable. 0 Disabled 1 Enabled
25 OF_EN1	FIFO 1 threshold Overflow interrupt enable. 0 Disabled 1 Enabled
26 EMPTY_EN1	FIFO 1 Empty interrupt enable. 0 Disabled 1 Enabled
27 FULL_EN1	FIFO 1 Full interrupt enable. 0 Disabled 1 Enabled
28 OR_EN0	FIFO 0 Overrun interrupt enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_FCR field descriptions (continued)

Field	Description
29 OF_EN0	FIFO 0 threshold Overflow interrupt enable. 0 Disabled 1 Enabled
30 EMPTY_EN0	FIFO 0 Empty interrupt enable. 0 Disabled 1 Enabled
31 FULL_EN0	FIFO 0 Full interrupt enable. 0 Disabled 1 Enabled

41.5.15 FIFO Threshold Register (CTU_FTH)

This register defines the threshold value for four FIFO modules. The threshold is used for comparison against the FIFO pointer and to evaluate if an overflow condition occurred for that FIFO. The overflow flag indicates that a certain amount of data, defined by the threshold value, is available in the FIFO for read. The overflow condition is set if the number of elements in the FIFO is greater than the threshold value. Thus, out of reset this register is set to detect one element in the respective FIFO since the initial value is zero.

NOTE

The threshold value should not exceed the FIFO size

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TH3								TH2								TH1								TH0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_FTH field descriptions

Field	Description
0–7 TH3	FIFO 3 Threshold.
8–15 TH2	FIFO 2 Threshold.
16–23 TH1	FIFO 1 Threshold.
24–31 TH0	FIFO 0 Threshold.

41.5.16 FIFO Status Register (CTU_FST)

This register has the flags that indicate the status of the FIFO modules. Up to four FIFOs are supported. FIFO overrun, overflow, empty, and full flags are available. If enabled by the FCR register bits, these flags can generate an interrupt to the CPU. There is one interrupt signal per FIFO which combines overrun, overflow, empty, and full flags.

NOTE

The only actual error indication corresponds to the overrun flags which stay asserted once the overrun condition occurs. They are cleared by writing 1 to the respective flag. The overflow, empty, and full flags correspond to the current state of the FIFOs.

NOTE

Bits OR3, OR2, OR1, and OR0 can be read and cleared by software

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OR3	OF3	EMP3	FULL3	OR2	OF2	EMP2	FULL2	OR1	OF1	EMP1	FULL1	OR0	OF0	EMP0	FULL0
W	w1c				w1c				w1c				w1c			
Reset	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0

CTU_FST field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 OR3	FIFO 3 Overrun interrupt flag. It is cleared if 1 is written to this bit location. 0 Interrupt has not occurred 1 Interrupt has occured
17 OF3	FIFO 3 threshold Overflow interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
18 EMP3	FIFO 3 Empty interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
19 FULL3	FIFO 3 Full interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
20 OR2	FIFO 2 Overrun interrupt flag. It is cleared if 1 is written to this bit location. 0 Interrupt has not occurred 1 Interrupt has occured
21 OF2	FIFO 2 threshold Overflow interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
22 EMP2	FIFO 2 Empty interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
23 FULL2	FIFO 2 Full interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
24 OR1	FIFO 1 Overrun interrupt flag. It is cleared if 1 is written to this bit location. 0 Interrupt has not occurred 1 Interrupt has occured
25 OF1	FIFO 1 threshold Overflow interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
26 EMP1	FIFO 1 Empty interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
27 FULL1	FIFO 1 Full interrupt flag.

Table continues on the next page...

CTU_FST field descriptions (continued)

Field	Description
	0 Interrupt has not occurred 1 Interrupt has occured
28 OR0	FIFO 0 Overrun interrupt flag. It is cleared if 1 is written to this bit location. 0 Interrupt has not occurred 1 Interrupt has occured
29 OF0	FIFO 0 threshold Overflow interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
30 EMP0	FIFO 0 Empty interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured
31 FULL0	FIFO 0 Full interrupt flag. 0 Interrupt has not occurred 1 Interrupt has occured

41.5.17 FIFO Right Aligned Data Register (CTU_FR*n*)

Address: 0h base + 80h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											ADC	N_CH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_FR*n* field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ADC	This bit indicates from which ADC Port the value in the DATA field corresponds to. 0 Data coming from ADC Port B 1 Data coming from ADC Port A
12–15 N_CH	Number of the channel that DATA field corresponds to.

Table continues on the next page...

CTU_FR*n* field descriptions (continued)

Field	Description
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA	Data of stored channel.

41.5.18 FIFO Signed Left Aligned Data Register (CTU_FL*n*)

NOTE

A read of this register consumes the data from the respective FIFO.

Address: 0h base + A0h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											ADC	N_CH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LA_DATA												0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_FL*n* field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ADC	This bit indicates from which ADC Port the value in the DATA field corresponds to. 0 Data coming from ADC Port B 1 Data coming from ADC Port A
12–15 N_CH	Number of the channel that DATA field corresponds to.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–28 LA_DATA	Data of stored channel.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.5.19 Error Flag Register (CTU_EFR)

The EFR register bits are cleared if a 1 is written to the bit position. Otherwise, the bit retains the previous value.

NOTE

> The ERRCMP field reads 0 at reset. When reset is released, this field remains cleared until the CTU timer clock is enabled. After the CTU timer clock is enabled, ERRCMP reads 1. This field must be cleared before enabling the interrupt requests.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7
Read	0		LIST_BE	CS	ET_OE	ERRCMP	T4_OE	T3_OE
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	T2_OE	T1_OE	ADC_OE	TGS_OSM	MRS_O	ICE	SM_TO	MRS_RE
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

CTU_EFR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 LIST_BE	List Busy Error. Indicates that a list of commands is trying to send commands to a busy ADC or to the same ADC at the same time. It also indicates a conflict in the self-test algorithm execution by the ADCs, see Self-Test Execution Errors section. Cleared if 1 is written to this bit location. 0 No error 1 Error
3 CS	Self test Counter Status. This bit indicates if a self-test algorithm is in progress. If 1 is written to this bit, a reset to the self test counter will occur, thus terminating the self-test algorithm execution. As a status bit it indicates: 0 All the counters are at reset value, thus none of the self-test algorithm are executing 1 One of the counters used in self test mode is not in the reset state, thus self-test is in progress.
4 ET_OE	External Trigger generation Overrun Error. This bit is set when an external trigger occurs while a trigger input is being processed by the TGS. It is cleared if 1 is written to this bit location. 0 No error 1 Error
5 ERRCMP	Error Compare. When a match occurs, the TGS Counter has reached the TGSCCR value. Cleared if 1 is written to this bit location.

Table continues on the next page...

CTU_EFR field descriptions (continued)

Field	Description
	0 No match 1 Match
6 T4_OE	Timer 4 trigger generation Overrun Error. This bit is set when a timer trigger input occurs while a trigger input is being processed by the TGS. Cleared if 1 is written to this bit location. 0 No error 1 Error
7 T3_OE	Timer 3 trigger generation Overrun Error. This bit is set when a timer trigger input occurs while a trigger input is being processed by the TGS. Cleared if 1 is written to this bit location. 0 No error 1 Error
8 T2_OE	Timer 2 trigger generation Overrun Error. This bit is set when a timer trigger input occurs while a trigger input is being processed by the TGS. Cleared if 1 is written to this bit location. 0 No error 1 Error
9 T1_OE	Timer 1 trigger generation Overrun Error. This bit is set when a timer trigger input occurs while a trigger input is being processed by the TGS. Cleared if 1 is written to this bit location. 0 No error 1 Error
10 ADC_OE	ADC command generation Overrun Error. This bit is set if an ADC command is issued when there is a command already being processed. Cleared if 1 is written to this bit location. 0 No Error 1 Error
11 TGS_OSM	The TGS Overrun in Sequential Mode. This bit is set when a new event occurs before the trigger event selected by the previous event had occurred. The error is generated only if the trigger from the TGS is linked to a SU output. Cleared if 1 is written to this bit location. 0 No overrun 1 Overrun
12 MRS_O	Master Reload Signal Overrun. If the time window defined by two consecutive MRS is not enough for the number of programmed events in the THCR registers, an error occurs and MRS_O flag is set. Cleared if 1 is written to this bit location. 0 No overrun 1 Overrun
13 ICE	Invalid Command Error. An error that occurs when the shared channel is accessed by two ADCs at the same time, or when an ADC is already using the channel. This situation can occur with a dual conversion command or with two single conversions in parallel mode. 0 No error 1 Error
14 SM_TO	Trigger Overrun (more than 8 ES) in TGS Sequential Mode. If the number of positive edges of ES is higher than 8, an error occurs and SM_TO flag is set. Cleared if 1 is written to this bit location. 0 No overrun 1 Overrun

Table continues on the next page...

CTU_EFR field descriptions (continued)

Field	Description
15 MRS_RE	Master Reload Signal Reload Error. If an MRS occurs while the user is updating a double-buffered register, the MRS_RE is set indicating an error condition. Cleared if 1 is written to this bit location. 0 No error 1 Error

41.5.20 Interrupt Flag Register (CTU_IFR)

The IFR register bits are cleared if a 1 is written to the bit position. Otherwise, the bit retains the previous value.

NOTE

The Tn_I fields read 0 at reset. When reset is released, these fields remain cleared until the CTU timer clock is enabled. After the CTU timer clock is enabled, the Tn_I fields read 1. These flags must be cleared before enabling the interrupt requests.

Address: 0h base + C2h offset = C2h

Bit	0	1	2	3	4	5	6	7
Read	0				SERR_B	SERR_A	ADC_I	T7_I
Write					w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	T6_I	T5_I	T4_I	T3_I	T2_I	T1_I	T0_I	MRS_I
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

CTU_IFR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 SERR_B	If this bit is set, it means that the time between the start of a conversion and the end of that conversion is out of the expected range which is defined by the EXPBR and CNTRNGR registers. See Expected Value B (EXPBR) and Counter Range (CNTRNGR) registers. Cleared if 1 is written to this bit location.
5 SERR_A	If this bit is set, it means that the time between the start of a conversion and the end of that conversion is out of the expected range defined by the EXPAR and CNTRNGR registers. See the Expected value A (EXPAR) and Counter Range (CNTRNGR) registers. Cleared if 1 is written to this bit location.
6 ADC_I	ADC command interrupt flag is set when a new command is issued. The interrupt related to this flag is enabled by the CIR bit in the CLR_x_n registers. Cleared if 1 is written to this bit location.

Table continues on the next page...

CTU_IFR field descriptions (continued)

Field	Description
7 T7_I	Trigger 7 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
8 T6_I	Trigger 6 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
9 T5_I	Trigger 5 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
10 T4_I	Trigger 4 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
11 T3_I	Trigger 3 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
12 T2_I	Trigger 2 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
13 T1_I	Trigger 1 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
14 T0_I	Trigger 0 interrupt flag is set when the corresponding trigger is issued. Cleared if 1 is written to this bit location.
15 MRS_I	MRS Interrupt flag is set when the Master Reload Signal occurs. The interrupt associated with this flag is enabled by the MRS_IE bit in the IR register. Cleared if 1 is written to this bit location.

41.5.21 Interrupt/DMA Register (CTU_IR)

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	T7_IE	T6_IE	T5_IE	T4_IE	T3_IE	T2_IE	T1_IE	T0_IE	0		SAF_CNT_B_EN	SAF_CNT_A_EN	DMA_DE	MRS_DMAE	MRS_IE	IEE
Write									▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_IR field descriptions

Field	Description
0 T7_IE	Trigger 7 Interrupt Enable. 0 Disabled 1 Enabled
1 T6_IE	Trigger 6 Interrupt Enable. 0 Disabled 1 Enabled
2 T5_IE	Trigger 5 Interrupt Enable. 0 Disabled 1 Enabled

Table continues on the next page...

CTU_IR field descriptions (continued)

Field	Description
3 T4_IE	Trigger 4 Interrupt Enable. 0 Disabled 1 Enabled
4 T3_IE	Trigger 3 Interrupt Enable. 0 Disabled 1 Enabled
5 T2_IE	Trigger 2 Interrupt Enable. 0 Disabled 1 Enabled
6 T1_IE	Trigger 1 Interrupt Enable. 0 Disabled 1 Enabled
7 T0_IE	Trigger 0 Interrupt Enable. 0 Disabled 1 Enabled
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SAF_CNT_B_EN	Enable the ADC Port B counter to check the conversion time. 0 Disabled 1 Enabled
11 SAF_CNT_A_EN	Enable the ADC Port A counter to check the conversion time. 0 Disabled 1 Enabled
12 DMA_DE	If this bit is set, a dma done acts as a write '1' in the GRE bit.
13 MRS_DMAE	If GRE bit is set, DMA request is issued on MRS occurrence.
14 MRS_IE	MRS Interrupt Enable. 0 Disabled 1 Enabled
15 IEE	Interrupt Error Enable. 0 Disabled 1 Enabled

41.5.22 Control ON Time Register (CTU_COTR)

Address: 0h base + C6h offset = C6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								COTGT							
Write	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_COTR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 COTGT	Control ON-Time and Guard Time for external trigger. See Scheduler subunit (SU) for a more detailed description.

41.5.23 Control Register (CTU_CR)

NOTE

The CTU_ADC_R bit resets all lists in execution. If CTU list execution is in parallel mode, this bit resets both lists even if they issued commands to different ADCs. Also, the self-test algorithm counters go to reset state.

NOTE

Bits Tn_SG, CTU_ADC_R, CGRE, and MRS_SG can be set by software. Bits CTU_ODIS, DFE, GRE, and TGSISR_RE can be read and set by software.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7
Read	▨	▨	▨	▨	▨	▨	▨	▨
Write	T7_SG	T6_SG	T5_SG	T4_SG	T3_SG	T2_SG	T1_SG	T0_SG
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	▨	CTU_ODIS	DFE	▨	FGRE	▨	GRE	TGSISR_RE
Write	CTU_ADC_R			CGRE	▨	MRS_SG		
Reset	0	0	0	0	0	0	0	0

CTU_CR field descriptions

Field	Description
0 T7_SG	Trigger 7 Software Generated. 0 S/W not generated 1 S/W generated
1 T6_SG	Trigger 6 Software Generated. 0 S/W not generated 1 S/W generated
2 T5_SG	Trigger 5 Software Generated. 0 S/W not generated 1 S/W generated
3 T4_SG	Trigger 4 Software Generated. 0 S/W not generated 1 S/W generated
4 T3_SG	Trigger 3 Software Generated. 0 S/W not generated 1 S/W generated
5 T2_SG	Trigger 2 Software Generated. 0 S/W not generated 1 S/W generated
6 T1_SG	Trigger 1 Software Generated. 0 S/W not generated 1 S/W generated
7 T0_SG	Trigger 0 Software Generated. 0 S/W not generated 1 S/W generated
8 CTU_ADC_R	CTU command list control state machine Reset. Self negated bit. This bit resets the command list in the CTU in case the ADC does not respond to a conversion command within the expected time window. After a reset, the command list stops execution and waits for a new trigger event in order to initiate a command list execution.
9 CTU_ODIS	CTU Output Disable. 0 Enabled 1 Disabled
10 DFE	Digital Filter Enable. 0 Disabled 1 Enabled
11 CGRE	Clear GRE to 0.
12 FGRE	Flag GRE. See Reload mechanism for more details.

Table continues on the next page...

CTU_CR field descriptions (continued)

Field	Description
13 MRS_SG	MRS Software Generated. 0 S/W not generated 1 S/W generated
14 GRE	General Reload Enable. See See Reload mechanism for more details. 0 Disabled 1 Enabled
15 TGSISR_RE	TGS Input Selection Register Reload Enable. Controls the re-load of the TGSISR register. This bit is self negated, thus it always returns zero. See Trigger Generator Subunit Input Selection Register (CTU_TGSISR) for more details.

41.5.24 Digital Filter Register (CTU_DFR)

Address: 0h base + CAh offset = CAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								FILTER_N							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_DFR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 FILTER_N	Digital Filter value. The external signal (EXT_IN) is considered at 1 if it is latched FILTER_N times at 1 and is considered at 0 if it is latched FILTER_N times at 0. If FILTER_N = 0, the filter is in bypass. 00000000 Filter is in bypass 00000001 In all combinations of FILTERN_N greater than 0, the filter is not in bypass -------- -- 11111111 In all combinations of FILTERN_N greater than 0, the filter is not in bypass

41.5.25 Expected Value A Register (CTU_EXPAR)

The EXPAR register has the expected number of system clock cycles (the slave bus clock) for an AD conversion to be completed by the ADC connected to Port A. If SAF_CNT_A_EN is set in the IR register and a conversion takes a number of cycles out of the interval defined by EXPA and CNTRNG to complete, then the SERR_A error bit is set in the IFR register. Please see the CNTRNGR for more details.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	EXPA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CTU_EXPAR field descriptions

Field	Description
0–15 EXPA	This value is the expected number of system clock cycles needed by ADC Port A to complete a conversion.

41.5.26 Expected Value B Register (CTU_EXPBR)

The EXPBR register has the expected number of system clock cycles (the slave bus clock) for an AD conversion to be completed by the ADC connected to Port B. If SAF_CNT_B_EN is set in the IR register and a conversion takes a number of cycles out of the interval defined by EXPB and CNTRNG to complete, then the SERR_B error bit is set in the IFR register. Please see the CNTRNGR for more details.

Address: 0h base + CEh offset = CEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	EXPB															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CTU_EXPBR field descriptions

Field	Description
0–15 EXPB	This value is the expected number of system clock cycles needed by ADC Port B to complete a conversion.

41.5.27 Counter Range Register (CTU_CNTRNGR)

The CNTRNGR register provides a way to mask out the less significant bits in the EXP_A and EXP_B fields of EXPAR and EXPBR registers, respectively. If set, bits in CNTRNG mask out related bits in EXP_A and EXP_B fields, thus providing a range in which a conversion should occur. The expression used for the conversion range check is: set SERR_A bit if (conversion counter & ~CNTRNG > EXP_A & ~CNTRNG) or (conversion counter & ~CNTRNG < EXPP_A & ~CNTRNG), where the conversion counter increments at system clock rate (slave bus clock rate). This equation also applies to SERR_B and EXP_B.

NOTE

The 1s in the CNTRNG must be set from the right to the left (less significant bits) without gap with 0.

NOTE

The safety counter allowed range is obtained by taking the EXPA/B value and exchanging to "X" the corresponding positions where the CNTRNG bit is 1.

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								CNTRNG							
Write	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_CNTRNGR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 CNTRNG	If set, the bits in this field mask out the equivalent bits of the expected values EXP_A and EXP_B. In this way, a range of system clock (slave bus clock) cycles is defined for a conversion to occur. This means the bit positions with 1s in the CNTRNG change the corresponding bits on EXPA/EXPB and safety counters A/B to "X". For example, with EXPA = 1b0010_1101 and CNTRNG = 0b0111, the safety counter can end with values of the format 0b0010_1XXX to avoid SERR_A error flag.

41.5.28 List Control/Status Register (CTU_LISTCSR)

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LIST1_BLK	0		LIST1_ADDR					LIST0_BLK	0		LIST0_ADDR				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															PAR_ LIST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTU_LISTCSR field descriptions

Field	Description
0 LIST1_BLK	List 1 Blocked flag. When set, this bit indicates that the address indicated by LIST1_ADDR had failed when trying to issue a command to the corresponding ADC. This bit is only meaningful if LIST_BE flag in EFR register is set. This bit is cleared when LIST_BE flag is cleared.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 LIST1_ADDR	List Address 1. This field shows the command address that was being executed when the LIST_BE flag in the EFR register was set.
8 LIST0_BLK	List 0 Blocked flag. This bit indicates that the address indicated by LIST0_ADDR had failed when trying to issue a command to the corresponding ADC. This bit is only meaningful if LIST_BE flag in EFR register is set. This bit is cleared when LIST_BE flag is cleared.

Table continues on the next page...

CTU_LISTCSR field descriptions (continued)

Field	Description
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 LIST0_ADDR	List Address 0. Indicates the command address being executed when LIST_BE flag in EFR register was set.
16–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PAR_LIST	Parallel mode List. This bit selects the parallel mode for the list execution. 0 List is executed in streaming mode 1 List is executed in parallel mode

41.6 Functional description

The CTU is comprised by two subunits: the Trigger Generator subunit, or TGS, and the Scheduler subunit, or SU. The TGS is responsible for generating triggers based on the module input signals connected to timer modules. The trigger signal from TGS is then connected to the scheduler subunit, which is responsible for the generation of ADC commands or triggers to on-chip logic such as timers. The following sections describe the TGS and SU in more detail.

41.6.1 Trigger Generator subunit (TGS)

The Trigger Generator subunit, TGS, generates up to eight trigger events. The main sub-blocks in the TGS are:

- Counter: generates sequential trigger events
- Trigger List: with 8 x 16-bit double-buffered registers for timing comparison

The TGS has two modes of operation:

- **Triggered mode**: each event source generates up to eight trigger event outputs.
- **Sequential mode**: each event source generates only one trigger event output.

Note

> An *event* means a pulse generated from the selected CTU trigger inputs. It is represented in Figure 41-5 by the ES signal (event signal).

External signals, represented by EXT_IN in Figure 41-5, may be asynchronous related to the CTU clock. For this reason the external trigger inputs are filtered by a CTU programmable digital filter. The external signal is considered at 1 if it is stable at 1 during FILTER_N CTU clock cycles and it is considered to be at 0 if it is stable at 0 during FILTER_N CTU clock cycles, where FILTER_N is defined in the digital filter register (DFR).

Trigger events can be initiated by hardware or by software. A software trigger is generated by writing to the CTU Control Register, CR. The CR register Tx_SG bits are automatically cleared by the CTU hardware when the respective trigger event is generated.

41.6.2 TGS in triggered mode

Figure 41-3 shows the TGS configuration in Triggered mode.

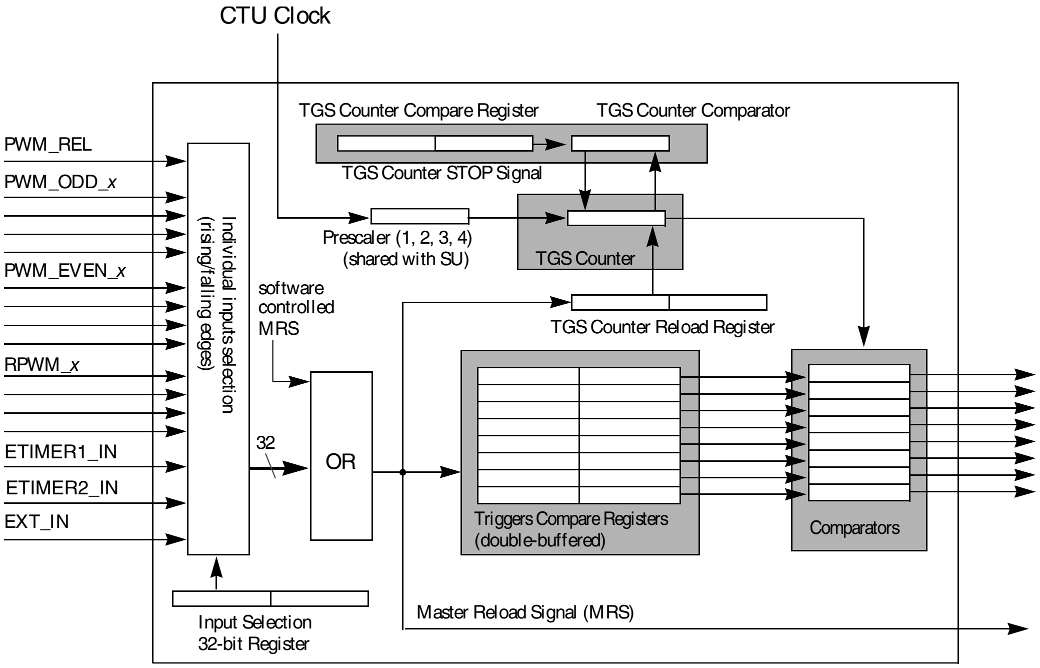


Figure 41-3. Trigger Generator subunit in triggered mode

Note

> The CTU clock should be connected to the same clock source used by the CTU inputs clock. In Figure 41-3 these inputs are from the PWM generator, eTimers, and EXT_IN.

The TGS has 16 input signals selected from the input selection register (TGSISR). The available selections are:

- Inactive, no selection
- Rising edge
- Falling edge
- Both edges

Up to 32 input events corresponding to 16 input signals are selected through the TGSISR register and OR-ed in order to generate the Master Reload Signal (MRS), which defines a *control cycle*.

The MRS can also be generated by software. The software operation is done by writing to the MRS_SG bit in the CTU Control Register. The MRS_SG bit is self cleared; thus, it reads always as 0. The MRS is used to re-load the TGS Counter Register with the values from the double-buffered TGSCRR register. The MRS is also used to re-load all double-buffered registers, such as the trigger compare registers, TGSCR and TGSCRR, with the corresponding buffered values.

The triggers list registers consist of 8 compare registers (T0CR through T7CR). Each register is associated with a comparator, where a match generates an output trigger. On the occurrence of a re-load triggered by MRS, the comparison for trigger events generation starts from the first cycle of counting. However, the comparison with the TGS Counter Compare Register (TGSCCR) is always disabled during one TGS clock cycle and then re-enabled. A match between TGSCCR and the TGS counter stops the TGS counter. Therefore, this one cycle disable allows the user to configure the TGSCCR and TGSCRR with the same value.

The prescaler for TGS and SU is defined by the PRES bits in the TGS Control Register and divides the CTU clock. The available selections for the prescaler values are 1, 2, 3, and 4.

Figure 41-4 shows a timing diagram for the TGS operation in triggered mode. The MRS and the trigger event occurrences are indicated along with the delay relative to the MRS occurrence.

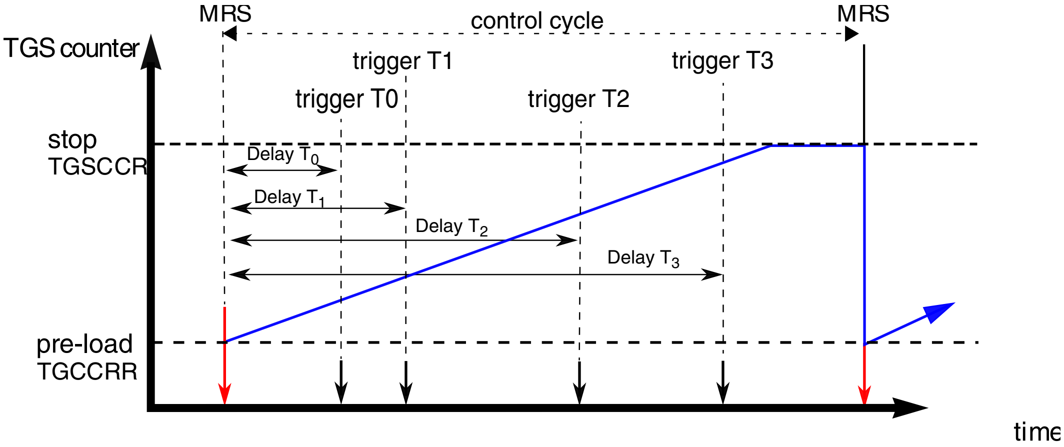


Figure 41-4. Example timing diagram for TGS in triggered mode

41.6.3 TGS in sequential mode

Figure 41-5 shows the blocks and signals related to the TGS in sequential mode.

In sequential mode only one of the 32 input signals is selected by the 5-bit MRS_SM (master reload selection). The selected signal re-loads the trigger list and resets the 3-bit ES counter which selects the trigger event. Sequences of up to eight trigger events are generated in one control cycle.

The 32 input events corresponding to 16 input signals with 2 edges detection per signal are individually enabled through the register TGSISR and OR-ed in order to generate the event signal ES. The ES signal is used to re-load the TGS counter register and to increment the 3-bit ES counter.

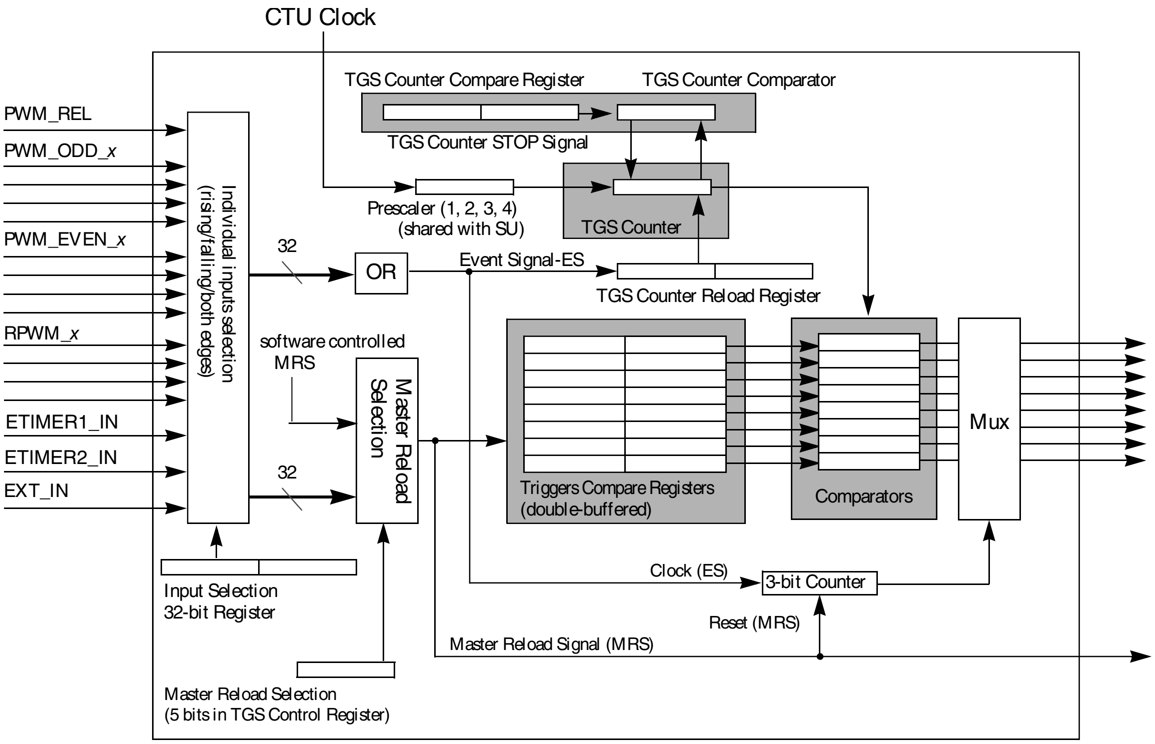

CTU Clock
TGS Counter Compare Register
TGS Counter Comparator
TGS Counter STOP Signal
PWM_REL
PWM_ODD_x
PWM_EVEN_x
RPWM_x
ETIMER1_IN
ETIMER2_IN
EXT_IN
Individual inputs selection (rising/falling/both edges)
32
OR
Prescaler (1, 2, 3, 4) (shared with SU)
TGS Counter
Event Signal-ES
TGS Counter Reload Register
software controlled MRS
Master Reload Selection
32
Triggers Compare Registers (double-buffered)
Comparators
Mux
Input Selection 32-bit Register
Clock (ES)
3-bit Counter
Reset (MRS)
Master Reload Signal (MRS)
Master Reload Selection (5 bits in TGS Control Register)

Figure 41-5. TGS in sequential mode

Figure 41-6 is a timing diagram for TGS in sequential mode showing the MRS and ES. The trigger events are indicated with the delay with respect to the ES. Note that initially ES and MRS are aligned. The TGS counter is re-loaded on each ES and starts counting up until the next ES or until it matches the value in TGSCCR register and stops.

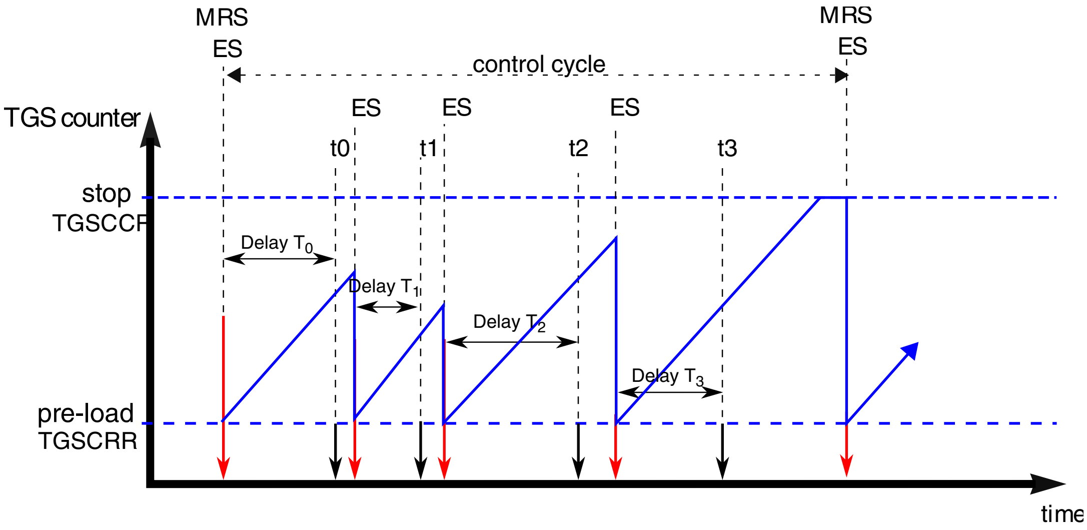


Figure 41-6. Example timing diagram for TGS in sequential mode

41.6.4 TGS counter

The TGS counter counts from negative to positive values as defined by the re-load and stop values defined in register TGSCRR and TGSCCR, respectively. The maximum counter value is 0x7FFF; after that, a counter wrap occurs and the counter value transitions from 0x7FFF to 0x8000. The following figure shows TGS counting examples.

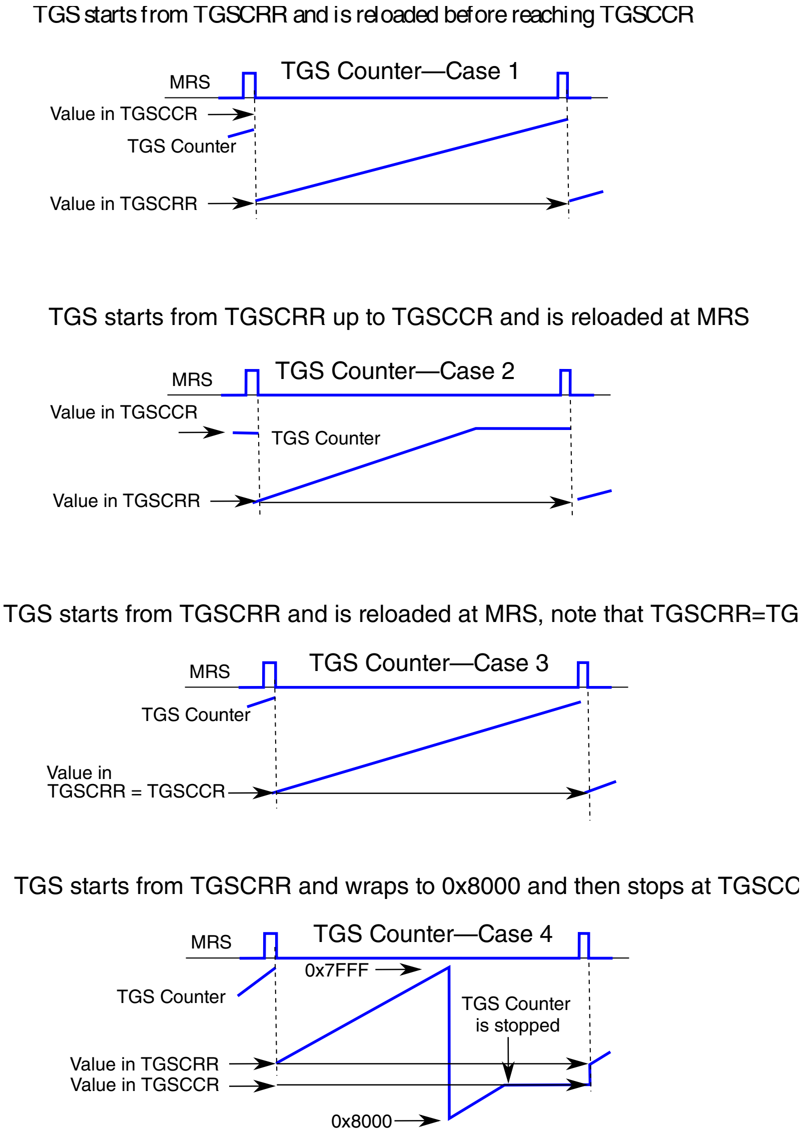


Figure 41-7. TGS counter behavior

Note

The TGS counter starts counting after an MRS and only stops if a counter wrap occurs and TGSCCR value is reached before the next MRS occurs.

41.6.5 Scheduler subunit (SU)

The following figure shows the block diagram of the Scheduler subunit. The SU trigger event outputs are:

1. ADC command or ADC stream of commands
2. eTimer1 pulse
3. eTimer2 pulse
4. External trigger pulse

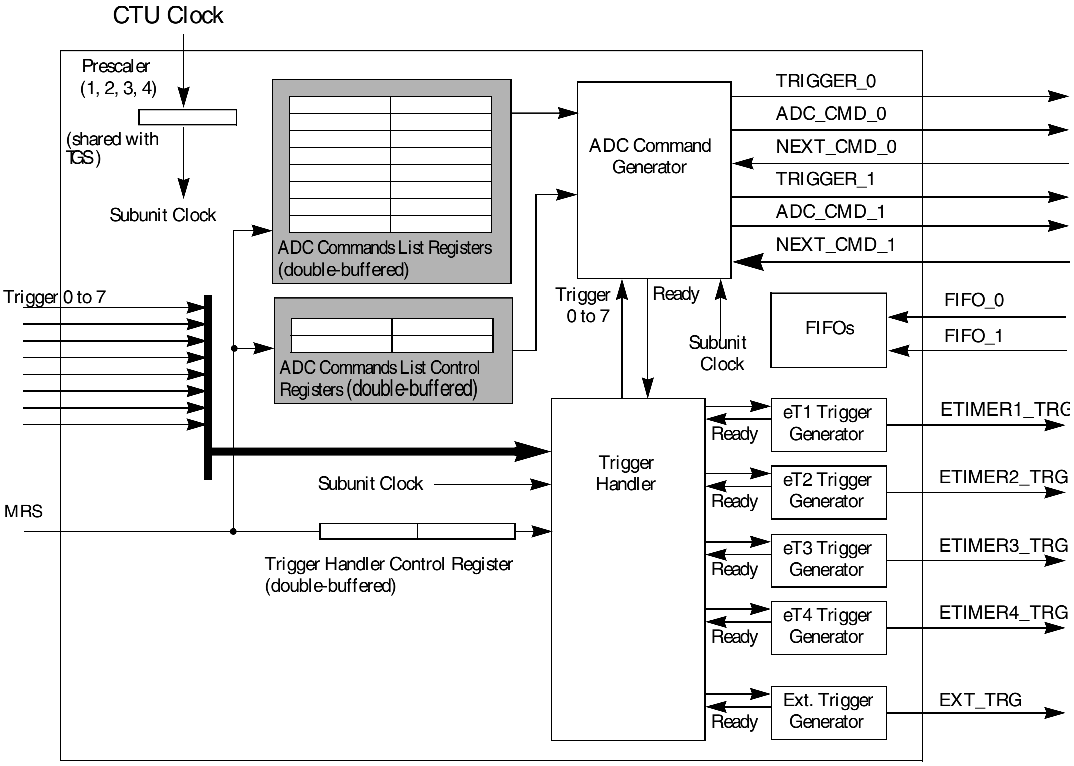


Figure 41-8. Scheduler subunit

The SU receives 8 trigger signals from the Trigger Generator subunit, TGS, and starts a command list to the selected ADC or generates the trigger event outputs. This module has the same functionality in both TGS modes: triggered and sequential. Each of the 4 SU outputs can be linked to any of the 8 trigger events from TGS. This is implemented by the Trigger Handler block. Each trigger event can be linked to one or more SU outputs.

If two events are linked at the same time to the same output (either ADC commands or output triggers) an error is generated and flagged in the EFR register's TnOE bits. In this case, only the trigger with the lowest index is used. For example, if trigger 0 and trigger 1 are linked to the same ADC output and they occur at the same time, an error is generated and the ADC command corresponding to trigger 0 is generated.

When a trigger is linked to an ADC, an associated ADC stream of commands is generated. The ADC Commands List Control Register CLCR1/2 defines the list entry point that is the first command to be executed. Subsequent list pointers are generated in an incremental way as the commands in the list are executed. When a trigger is linked to an eTimer or an EXT_TRG shown in the above figure, an event is generated on the corresponding output. The external trigger output has two modes of operation configured by TGSCR register bit ET_M:

- Pulse mode: a pulse with one system clock width is generated at the output pin
- Toggle mode: the output pin toggles its state as a result of each trigger event

An *ON-Time* and *Guard-Time* are defined for both Pulse and Toggle modes. The COTR register, Control *On-Time* Register, defines the (*ON-Time* + *Guard-Time)* duration. After a trigger has occurred a new trigger will be generated only after (*ON-time* + *Guard-Time)*. The COTR register only applies to external triggers; it does not affect ADC commands.

41.6.6 ADC commands list

The SU implements a command list that stores the ADC commands. The command list stores up to 24 x 16-bit commands (see ADC command list format) in a double-buffered implementation. The command list buffer registers may be updated at any time between two consecutive MRS, but the changes are transferred from the buffers to the actual registers only after an MRS occurs, so a correct reload procedure is performed (see Reload mechanism). The first command in a list is pointed by the 5-bit CLCRx register. Once a command list is triggered, it executes until the last command is found. The last command (n) is defined by the LC bit of the next command (n+1) in the list. Note that the command with LC=1 does not belong to the list and thus it is not executed. Note also that the LC bit is ignored in the first command of a list. See the command list registers, CLR_A_n, CLR_B_n, or CLR_C_n, where n = 1,….,24. The figure below describes the state diagram for the Commands List execution. A sequence of commands on the list is executed until the last command is found. See ADC command list format for more details.

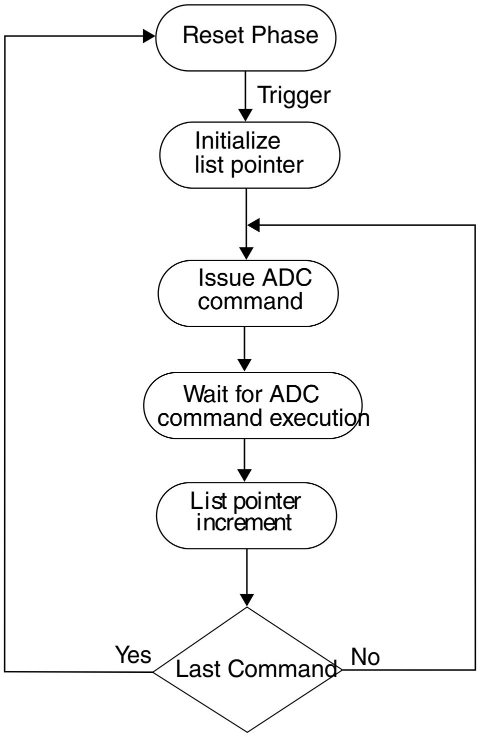


Figure 41-9. ADC command list scanning state diagram

Note

The CTU reads the next command line (the LC bit) to determine if the present command is the last one to be executed.

41.6.7 ADC command list format

The ADC command can be configured for single or dual conversion. In single conversion mode, the command targets one ADC only. In dual conversion mode, the conversion command is sent to both ADCs at the same time. Due to the clock domain crossing between CTU and ADCs, there could be a misalignment of up to one ADC clock cycle between the two commands.

The result of each conversion, in both modes, is stored in one of the four available FIFOs. In dual conversion mode, both ADCs store the result of their conversion in the same FIFO. If both ADCs access the FIFO in the same clock cycle, ADC unit A has higher priority, otherwise the first available conversion result is stored first.

The CTU supports up to 32 channels for both ADCs. The same physical channel cannot be assigned for both ADCs in a dual conversion mode. If there is an attempt to do so, an error is flagged in the EFR register's ICE bit, but the CTU sends the command to both ADCs.

In general, an ADC command is composed by the following fields depending upon the conversion format used (see Commands List Register A for ADC single-conversion mode commands (CTU_CLR_A_*n*)):

- Channel A: ADC A channel number (4 bits)
- Channel B: ADC B channel number (4 bits)
- Target ADC selection: ADC A or ADC B used in single conversion mode only (1 bit)
- FIFO selection bits for the selected ADC unit: up to four FIFOs (2 bits)
- Conversion mode selection: single or dual conversion mode (1 bit)
- Last command (LC): defines the last command in a list (1 bit)
- Interrupt request: enable interrupt request on command execution (1 bit)

The first command of a list is indicated by the CLCR1/2 registers' Tx_INDEX field. The last command is indicated by the LC (Last Command bit) of the following command in the command list. The first command is always executed. The execution of the following commands is controlled by the LC bit.

The figure below describes several scenarios for the command sequencing. If the first command indicated by the CLCR1/2 register is one of the lines identified as "Last command," then the list is a single command list. Note also that the list pointer wraps back to the first command in the list after command 24 is executed and if it is not the last command. If no command is defined as the last command, then the list will be executed continuously in a loop of 24 commands.

- Command List A: two command queues are available with elements 4 and 24 being last commands
- Command List B: sub-list 1 with one single command at address 1, and sub-list 2 with a queue of commands with last command at address 24
- Command List C: sub-list 2 with one single command at address 24, and sub-list 1 with a command queue with last command at address 23
- Command List D: two command queues with a wrap at address 24

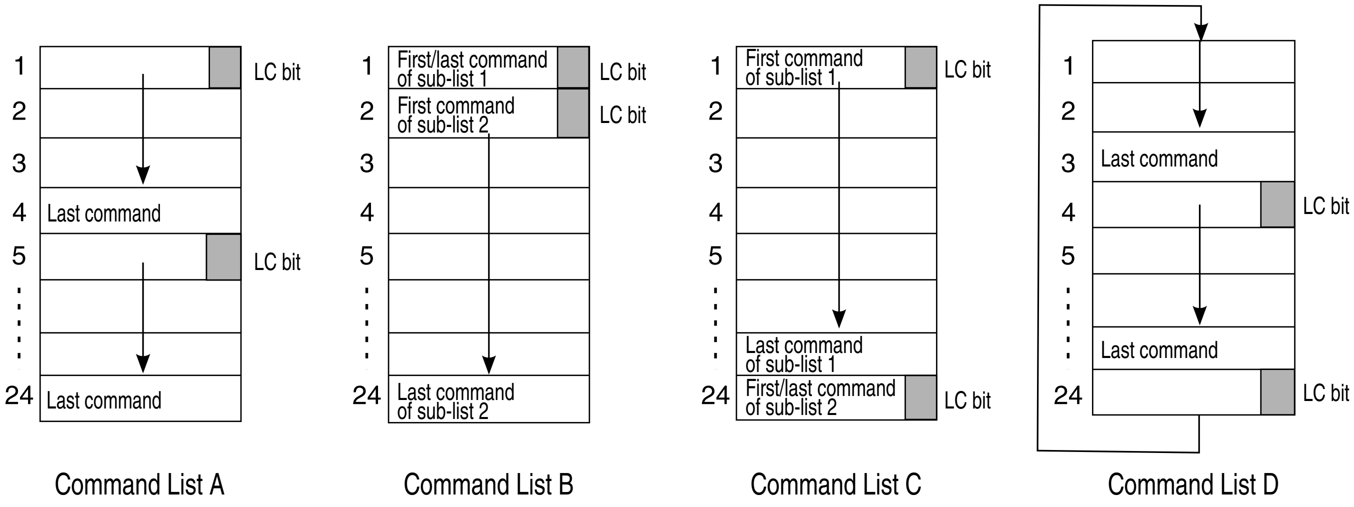

1
2
3
4
5
24
Last command
Last command
LC bit
LC bit
Command List A
First/last command of sub-list 1
First command of sub-list 2
Last command of sub-list 2
LC bit
LC bit
Command List B
First command of sub-list 1
Last command of sub-list 1
First/last command of sub-list 2
LC bit
LC bit
Command List C
Last command
Last command
LC bit
LC bit
Command List D

Figure 41-10. Command list sequencing

Note

> The addresses shown in in the above figure do not represent addresses in the memory map. Please refer to the Memory Map section to check for the actual addresses.

41.6.8 ADC command list operation modes

There are two modes of operation regarding the ADC command list execution: ***streaming*** mode and ***parallel*** mode. In streaming mode, the command lists should behave as a stream of commands, meaning that two or more lists cannot be executed at the same time, but only in a sequence. Thus if list 1 is triggered, it should finish the execution of its last command before list 2 can be triggered. If there is an attempt to trigger more than one list at the same time, then an error occurs.

In parallel mode, up to two lists can be executed at the same time. In order to avoid errors during the execution of two parallel lists, they should not have commands for both ADCs. Therefore, if list 1 has commands for ADC A, then list 2 should only have commands for ADC B. Note also that if a shared analog input is used by one of the lists, it should not be used by the other list in order to avoid a contention and therefore an error. ADC dual conversion commands should also not be used in parallel mode.

Figure 41-11 shows the execution of two lists in streaming mode. While a list is being executed, any attempt to trigger the execution of another list while in this mode will cause an error. Note that both lists may have commands for ADC A or B with no restriction. Also, dual commands may be used.

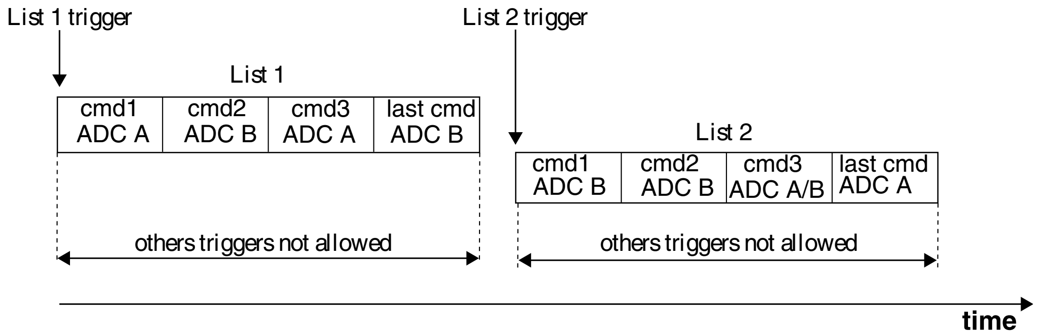


Figure 41-11. List execution in Streaming Mode

Figure 41-12 shows the execution of two lists in parallel mode. There are no restrictions for the triggers of both lists. After the two lists are triggered, no other triggers are allowed. If there is an attempt to trigger a third list while lists 1 and 2 are executing, then an error will be issued by setting ADC_OE bit in EFR register. Note that list 1 only has commands for ADC A and list 2 only has commands for ADC B. Having ADC A and ADC B commands in the same list may cause an error since two commands for the same ADC could be executed from different lists at the same time.

Note

The mixing of commands for both ADCs in the same list in parallel mode must not be used since the correct execution of commands in this case is not guaranteed by the CTU hardware.

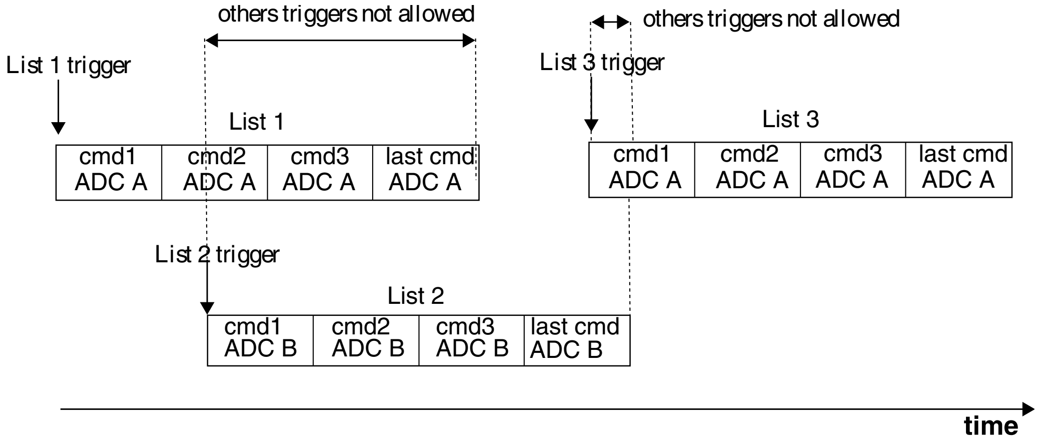


Figure 41-12. List execution in parallel mode

The list mode is controlled by the PAR_LIST bit in the LISTCSR register.

41.6.9 ADC results FIFOs

ADC results are stored in one of the four FIFOs. The FIFOs allow the storage of ADC results according to the type of data acquisition (phase currents, rotor position, ground-noise, and so on) so the software can distinguish data from several ADC channels. Each FIFO has its own interrupt line, DMA request signal, and a status register. The target FIFO for a conversion result is specified in the ADC command. Each FIFO element is 32 bits wide. The following figure shows the FIFO selection for ADC A and ADC B results.

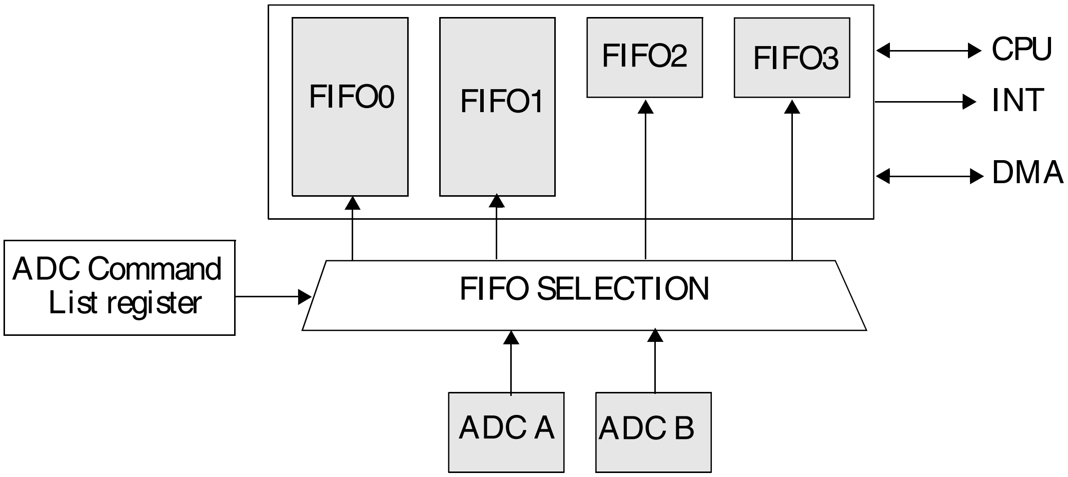


Figure 41-13. FIFO selection for ADC results storage

The CTU FIFOs are fixed depth. For example:

- FIFO0 and FIFO1: 16 entries each, dimensioned for full PWM period current acquisitions
- FIFO2 and FIFO3: 4 entries each, dimensioned for low rate acquisitions

FIFO access can be done by 16-bit or 32-bit read transaction. In 32-bit reads the conversion result, the target ADC, and the channel number are accessed. 8-bit access must not be used; otherwise, the FIFO pointer increments and the complete data is not retrieved. The FIFO result register can be read in a right- or left-aligned format using two different addresses:

- Unsigned right-justified, read from register FLx (for FIFO x)

  (Conversion result is unsigned right-justified data, i.e. bits [11:0] are used for 12-bit resolution and bits [15:12] always return zero)

- Signed left-justified, read from register FLx (for FIFO x)

(Conversion result is signed left-justified data, i.e. bit [15] is reserved for sign and is always read as zero, bits [14:3] are used for 12-bit resolution and bits [2:0] always return zero)

The FIFO access should be based on the status bits in the FCR register. Empty and full indication are provided. If data is read from an empty FIFO, then the read data should not be considered as valid data. Note that there is no indication of an underflow condition. If the FIFO is full and there is an attempt to write more data, then the new data is not written to the FIFO and an overrun flag is set.

41.6.10 Reload mechanism

In order to assure coherent programming, several CTU registers are double-buffered, thus allowing the programming of new data on the buffers while not affecting data being used. All registers are uploaded with buffer data at the same time. For the majority of the CTU registers the re-load is controlled by the Master Reload Signal, MRS.

Since the Master Reload Signal is generated by the hardware, it may occur while the software is still updating the buffer registers. In this case, noncoherent values are written to the registers because the CPU did not finish the programing of all registers. In order to avoid this situation, a General Reload Enable, GRE, control bit is provided. If GRE is cleared then no reload occurs. If this bit is set, then the reload is done when MRS occurs. The GRE bit is automatically cleared by MRS or is cleared by software using the CGRE bit in the same register. The CGRE bit is self-cleared and thus always read as zero.

The software should be able to program all registers buffers within a control cycle. In order to help the software to evaluate that a flawless register programing was executed, the MRS_RE flag is provided in the EFR register. If cleared, this flag indicates that all register buffers were programed before MRS occurred, as follows:

- If any of the double buffered registers is written, the FGRE flag is set indicating that a programing cycle has initiated.
- Set GRE to indicate that all updates to the buffer registers were performed.
- The reload of all registers is executed when the MRS occurs. Since GRE is set then MRS_RE will not be set.
- GRE and FGRE are cleared when MRS occurs.

See Figure 41-14 for more details.

An error is flagged by MRS_RE when MRS occurs, GRE is cleared, and FGRE is set. This scenario indicates that the software has initiated a register programming but has not finished it before the MRS, as follows:

- If any of the double buffered registers is written, the FGRE flag is set indicating that a programing cycle has initiated.
- MRS occurs and GRE is not set. In this case MRS_RE is set and, if enabled, an interrupt is generated.
- FGRE is not cleared when MRS occurs and no register is updated, since GRE is cleared.

Note that the software does not need to update all double-buffered registers during a *control cycle*. In order to avoid the error indication it is sufficient to set GRE before a MRS occurs.

A race condition occurs when GRE is being cleared by CGRE at the same time that an MRS occurs. In this case, GRE is considered to be cleared and no reload occurs. Another race condition may occur between CGRE and GRE being set. Since these two bits are in the same register they could eventually be set at the same time. In this case, CGRE has precedence; thus, GRE is cleared.

Note

> MRS has priority over the TGSISR_RE bit in the CR register, which is the re-synchronization bit of the TGSISR.

The following figure describes two scenarios for the register re-load. Scenario (a) is a normal re-load where the buffers of the double-buffered registers are programmed and the GRE bit is set before the next MRS occurs. The scenario presented in (b) describes an error condition where the software did not signal the hardware that all registers are ready for re-load before the next MRS. This is indicated by GRE being at zero when MRS occurs. In this case, MRS_RE flag is set indicating the fault condition.

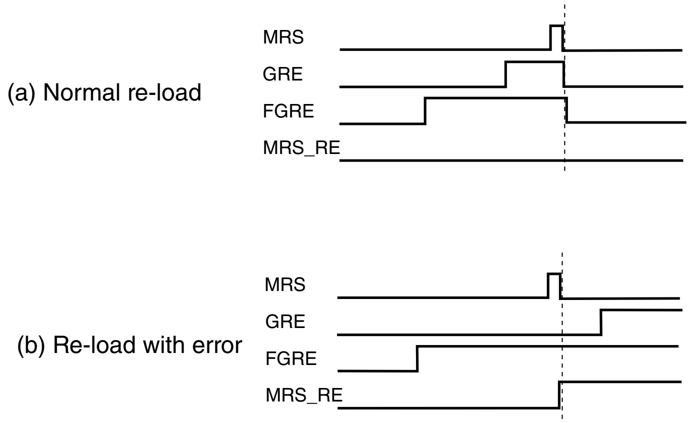


Figure 41-14. Reload scenarios

41.7 Interrupts and DMA Requests

41.7.1 DMA support

The DMA interface can be used to configure the CTU registers. A DMA request is issued if MRS occurs and the GRE bit is set. In this case, the previously transferred data is loaded from the buffers into the registers and a new transfer cycle is initiated with the DMA request. The DMA done signal from the DMA controller is used to set GRE bit which allows the reload to be performed. The following figure describes this operation.

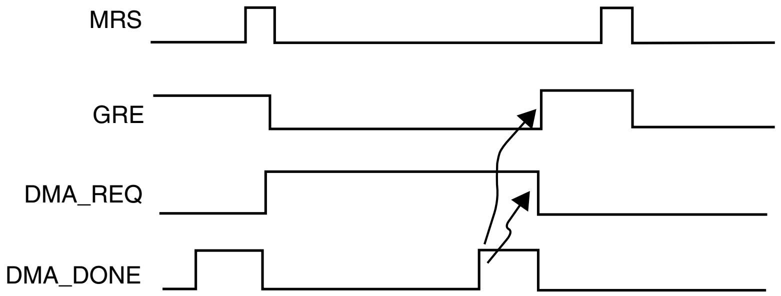


Figure 41-15. Register reload using DMA

DMA support can also be provided for reading FIFO stored data. Each FIFO can be configured to perform a DMA request when the number of stored words reaches a threshold value defined in the FTH register. See the FIFO Threshold, FTH, register for more information. After a DMA_DONE and the remaining data in the FIFO is below the watermark, then the DMA request is removed. Thus for an efficient operation, the DMA should be configured to execute a loop reading all data in the FIFO considering the amount of data the same as defined by the watermark.

41.7.2 CTU faults and errors

The CTU detects and signals the following error conditions:

- MRS reload error: MRS occurs while user is updating the double-buffered registers.
- Sequential mode trigger overrun SM_TO: more than 8 event signals occur before the next MRS in TGS sequential mode.

- Trigger event overrun indicated by 4 interrupt signals ADC_OE, T0_OE, T1_OE and ET_OE: a trigger event occurs during the time frame when the previous trigger was not completed.
- The CTU allows the user to pre-set a trigger to the eTimer1 in the middle of an ADC conversion. In parallel mode even two ADCs' commands can be issued at the same time. The error only occurs if the second trigger is targeting the same ADC.
- Invalid (unrecognized) ADC command occurs and the ICE bit is set.
- Master reload signal overrun MRS_O: MRS occurs before all triggers selected by (Tn_E and (TnETE or TnTmE or Tn_ADCE)) bits in THCR1/2 registers have occurred.
- TGS_OSM TGS overrun in sequential mode: a new event occurs before the trigger event selected by the previous event had occurred.

The faults/errors flags in the CTU error flag register and in the CTU interrupt flag register can be cleared by writing 1 to the respective bit. Writing 0 to these bits has no effect.

Note

> The CTU does not support write-protection mechanism for any register, meaning that all registers are accessible at any time.
>
> The CTU does not generate a transfer error when its reserved address space is accessed.

41.7.2.1 CTU parallel list errors

The command list operation in parallel mode may generate errors since two lists may be in execution at the same time. Below some error conditions are described.

Commands for the same ADC are issued at the same time by two lists. In this case:

- One of the commands is disregarded and an error is issued.
- The LIST_BE bit in the EFR register is set.
- The address of the list commands are shown in the LISTCSR register.
- The address of the command that was not executed is indicated in the LISTCSR register.

A third list is triggered when two lists are already being executed. In this case:

- The third list is not executed and an error is indicated by the ADC_OE bit in the EFR register.

One list is being executed and a second list is triggered; the first command of the second list is for the same ADC being used by the first list. In this case:

- The command of the second list is not executed.
- The address of the second list command is shown in the LISTCSR register.
- The LIST_BE bit in the EFR register is set.

41.7.3 CTU interrupt/DMA requests

The CTU generates the following interrupt/DMA requests which are controlled by the IR register:

- Error interrupt request, see CTU faults and errors (1 interrupt line)
- ADC command interrupt request (1 interrupt line).
- MRS interrupt request (1 interrupt line)
- Trigger event interrupt request (1 interrupt line for each trigger event)
- FIFOs interrupt and/or DMA transfer request (1 interrupt line for each FIFO). See FIFO control register, FCR.
- DMA transfer request on the MRS occurrence if GRE bit is set

41.8 Self test mode

The CTU is capable of issuing a self test command to the ADC.

The CLRx register is used to set up a self test command. Table 41-2 describes the ST0 and ST1 bits that control self-test functionality.

Table 41-2. Self-test command

ST_CMS bit	ST1 bit	ST0 bit	Command description
0	Not Used	0	no self test single conversion
1	Not Used	0	no self test dual conversion
Not Used	0	1	self test command

Table continues on the next page...

Table 41-2. Self-test command (continued)

ST_CMS bit	ST1 bit	ST0 bit	Command description
0	1	1	no self test command single conversion
1	1	1	no self test command dual conversion

Two algorithms are available in self test mode, see Table 41-3.

Table 41-3. Self-test algorithm selection

ALG[1]	Number of steps	Executed Algorithm
00	3	Algorithm S
01[2]	–	Reserved
10	12	Algorithm C
11	15	Algorithm FULL (S + C)

[1] ALG refers to the CLRx register field.
[2] The 01 selection causes a command execution error and the LIST_BE flag is set in the Error Flag Register (EFR).

The self-test S algorithm is executed in an atomic burst of conversions independent of the BSIZE value in the CLRx register. Below are described the steps for this algorithm execution.

- Step 0: VBGAP/VREF test
- Step 1: VDD/VREF test
- Step 2: VREF/VREF test

When the FULL algorithm is selected, the CTU executes the S algorithm followed by the C algorithm.

The self-test execution follows the trigger scheme selected for the CTU, either triggered mode or sequential mode can be selected.

In order to completely execute one of the algorithms described in Table 41-3 a certain number of steps need to be performed. Each step corresponds to the time for one ADC conversion to complete. These steps may be executed at once, by just one command, or they can be interleaved with other ADC commands. In order to executed self-test commands interleaved with other conversion commands it is sufficient to set the BSIZE field in the CLRx register for a value lower than the number of steps defined for the selected algorithm. The following figure shows the execution of the C algorithm interleaved with regular ADC commands.

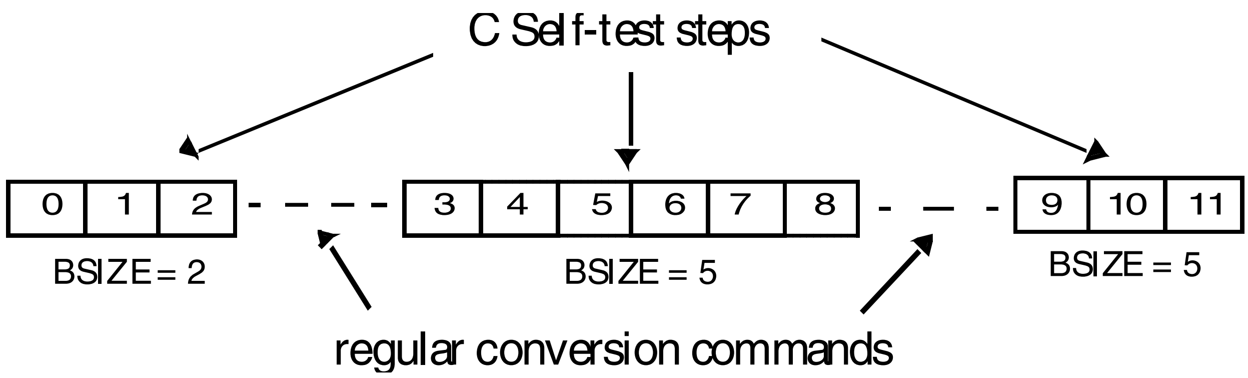


Figure 41-16. C Algorithm execution interleaved with others ADC commands

In the figure above, the last C self-test sequence intentionally shows BSIZE set to a larger value than the remaining steps needed to complete the C algorithm. In this case, the number of executed self-test steps does not follow BSIZE but instead it follows the number of remaining steps for the selected self-test algorithm, which in this case is 3 steps.

Note

> The S algorithm steps do not interleave with other ADC commands even if BSIZE is smaller than 3. Independent of BSIZE value all the 3 steps of the S algorithm are executed in a contiguous sequence.

The number of steps needed to completely execute the selected self-test algorithm is as follows:

- S algorithm requires 3 steps
- C algorithm requires 12 steps
- FULL runs all algorithm and requires 15 steps

In the figure below, the FULL algorithm is executed interleaved with other ADC commands. The BSIZE = 3 in the first and second self-test commands produces different number of executed self-test steps.

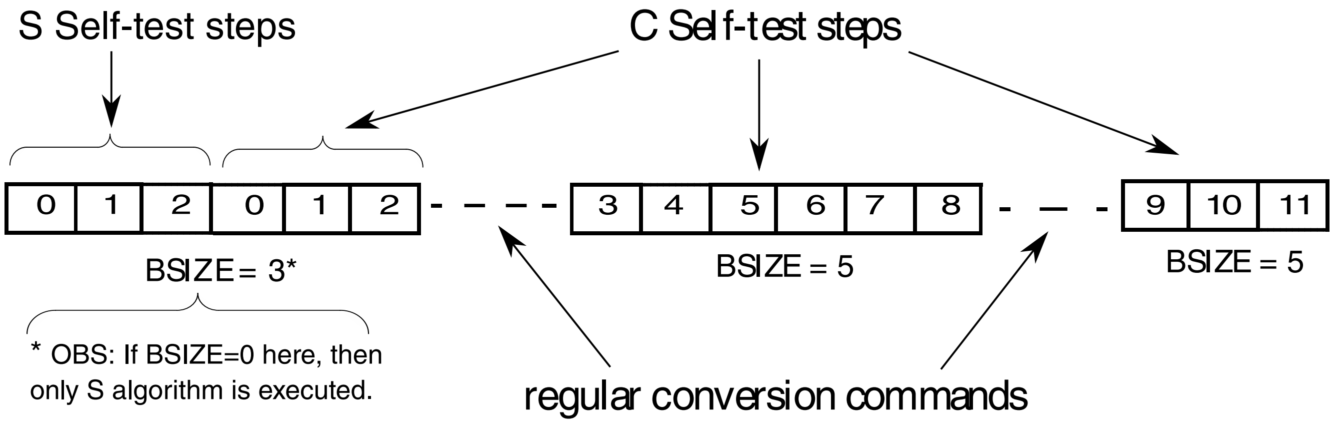

S Self-test steps
C Self-test steps
0
1
2
0
1
2
3
4
5
6
7
8
9
10
11
BSIZE= 3*
BSIZE = 5
BSIZE = 5
* OBS: If BSIZE=0 here, then only S algorithm is executed.
regular conversion commands

Figure 41-17. FULL algorithm execution interleaved with other ADC commands

41.8.1 Self-test execution errors

When a self-test algorithm is initiated for an ADC, it should complete before any other self-test algorithm is started in the same ADC. In other words, one self-test algorithm should not be interleaved with another self-test algorithm. Any attempt to do so will cause the ongoing list execution to be halted and cause the LIST_BE error flag (see LIST_BE) to be set. In order to evaluate if a self-test algorithm is running, the CS bit (see CS) can be checked. If set, this bit indicates a self-test has initiated in one of the ADCs. If a self-test algorithm conflict occurs, as described, the recovery procedure is to issue a software reset through the CTU_ADC_R bit in the CR register.

An error condition (see LIST_BE) is also caused when the reserved value 01 is selected for the self-test algorithm.

41.9 Power saving modes

In order to reduce power consumption, a STOP mode is provided.

41.9.1 STOP mode

The STOP signal is an input to the CTU that disables the clocks for all registers including memory mapped registers. Thus, writes have no effect and reads do not return valid data after STOP mode is entered.

The CTU enters STOP mode immediately after STOP signal is asserted, thus in order to avoid wrong module operation some housekeeping should be performed prior to entering low power mode:

- The triggers should be disabled thus no pending ADC commands exist.
- The CTU output need also be disabled through the CTU_ODIS bit.
- The FIFOs should be empty with no pending data conversion for read.

Chapter 42
System Timer Module (STM)

42.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

This section provides an overview, list of features, and modes of operation for the STM.

42.1.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

42.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

42.1.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

42.2 External signal description

The STM does not have any external interface signals.

42.3 Memory map and registers

The STM programming model includes a group of 32-bit registers—a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

STM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STM Control Register (STM_CR)	32	R/W	0000_0000h	42.3.1/1275
4	STM Count Register (STM_CNT)	32	R/W	0000_0000h	42.3.2/1276
10	STM Channel Control Register (STM_CCR0)	32	R/W	0000_0000h	42.3.3/1276
14	STM Channel Interrupt Register (STM_CIR0)	32	w1c	0000_0000h	42.3.4/1277
18	STM Channel Compare Register (STM_CMP0)	32	R/W	0000_0000h	42.3.5/1277
20	STM Channel Control Register (STM_CCR1)	32	R/W	0000_0000h	42.3.3/1276
24	STM Channel Interrupt Register (STM_CIR1)	32	w1c	0000_0000h	42.3.4/1277
28	STM Channel Compare Register (STM_CMP1)	32	R/W	0000_0000h	42.3.5/1277
30	STM Channel Control Register (STM_CCR2)	32	R/W	0000_0000h	42.3.3/1276
34	STM Channel Interrupt Register (STM_CIR2)	32	w1c	0000_0000h	42.3.4/1277
38	STM Channel Compare Register (STM_CMP2)	32	R/W	0000_0000h	42.3.5/1277
40	STM Channel Control Register (STM_CCR3)	32	R/W	0000_0000h	42.3.3/1276
44	STM Channel Interrupt Register (STM_CIR3)	32	w1c	0000_0000h	42.3.4/1277
48	STM Channel Compare Register (STM_CMP3)	32	R/W	0000_0000h	42.3.5/1277

42.3.1 STM Control Register (STM_CR)

The STM Control Register includes the prescale value, freeze control and timer enable bits.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPS								0						FRZ	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2 ----------------------- 0xFF Divide system clock by 256
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. **NOTE:** When the MCU enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
31 TEN	Timer counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

42.3.2 STM Count Register (STM_CNT)

The STM Count Register holds the timer count value.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CNT field descriptions

Field	Description
0–31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

42.3.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCRn) has the enable bit for channel n of the timer.

Address: 0h base + 10h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CCR*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEN	Channel Enable 0 The channel is disabled. 1 The channel is enabled.

42.3.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register has the interrupt flag for channel n of the timer.

Address: 0h base + 14h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CIR*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

42.3.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Address: 0h base + 18h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CMP*n* field descriptions

Field	Description
0–31 CMP	Compare value for channel n. If the STM_CCRn[CEN] bit is set and the STM_CMPn register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIRn[CIF] bit is set.

STM_CMP*n* field descriptions (continued)

Field	Description

42.4 Functional description

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the system clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode, the counter continuously increments. When enabled in debug mode, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode; otherwise, it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*), and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit. When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

NOTE

> The STM counter does not advance when the system clock is stopped.

Chapter 43
Software Watchdog Timer (SWT)

43.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

This section provides an overview, list of features, and modes of operation for the SWT.

43.1.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires, the SWT generates an interrupt or hardware reset request. The SWT can be configured to generate a reset request or interrupt on an initial time-out. The SWT always generates a reset request on a second consecutive time-out.

43.1.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode

- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

43.1.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit in the SWT_CR. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

43.2 External signal description

The SWT module does not have any external interface signals.

43.3 Memory Map and Registers

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_CR are set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

The SWT memory map is shown in the following table.

SWT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SWT Control Register (SWT_CR)	32	R/W	FF00_011Bh	43.3.1/1281
4	SWT Interrupt Register (SWT_IR)	32	R/W	0000_0000h	43.3.2/1284

Table continues on the next page...

SWT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8	SWT Time-out Register (SWT_TO)	32	R/W	0003_FDE0h	43.3.3/1284
C	SWT Window Register (SWT_WN)	32	R/W	0000_0000h	43.3.4/1285
10	SWT Service Register (SWT_SR)	32	W	0000_0000h	43.3.5/1285
14	SWT Counter Output Register (SWT_CO)	32	R	0000_0000h	43.3.6/1286
18	SWT Service Key Register (SWT_SK)	32	R/W	0000_0000h	43.3.7/1287

43.3.1 SWT Control Register (SWT_CR)

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD		RIA	WND	ITR	HLK	SLK	Reserved	STP	FRZ	WEN
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1

SWT_CR field descriptions

Field	Description
0 MAP0	Master Access Protection for Master 0. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
1 MAP1	Master Access Protection for Master 1. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
2 MAP2	Master Access Protection for Master 2. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
3 MAP3	Master Access Protection for Master 3. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
4 MAP4	Master Access Protection for Master 4. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
5 MAP5	Master Access Protection for Master 5. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
6 MAP6	Master Access Protection for Master 6. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
7 MAP7	Master Access Protection for Master 7. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
8–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 SMD	Service Mode. 00 Fixed Service Sequence, the watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to the SWT_SR. 01 Keyed Service Sequence, the watchdog is serviced by writing two pseudorandom key values to the SWT_SR. 10 Reserved—do not use. Writing a value can cause the SWT not to be serviced. 11 Reserved—do not use. Writing this value can cause the SWT not to be serviced.

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
23 RIA	Reset on Invalid Access. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a system reset if WEN=1
24 WND	Window Mode. 0 Regular mode, service sequence can be done at any time 1 Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
25 ITR	Interrupt Then Reset **NOTE:** For a description of how this chip implements SWT reset requests resulting from SWT time-outs, see the chip-specific SWT information. 0 Generate a reset request on a time-out 1 Generate an interrupt on an initial time-out; generate a reset request on a second consecutive time-out
26 HLK	Hard Lock. This bit is cleared only at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock. This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
28 Reserved	Reserved This field is reserved.
29 STP	Stop Mode Control. Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control. Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled. 0 SWT is disabled 1 SWT is enabled

43.3.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_IR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request 1 Interrupt request due to an initial time-out

43.3.3 SWT Time-out Register (SWT_TO)

The SWT Time-out (SWT_TO) register contains the 32-bit time-out period. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WTO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0

SWT_TO field descriptions

Field	Description
0–31 WTO	Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with 261600 cycles, when the service sequence is written or when the SWT is enabled.

43.3.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_WN field descriptions

Field	Description
0–31 WST	Window Start Value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

43.3.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																	WSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_SR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the soft lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] field is 01b, two pseudorandom key values are written to service the watchdog, see Functional description for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

43.3.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_CO field descriptions

Field	Description
0–31 CNT	Watchdog Count. When the watchdog is disabled (SWT_CR[WEN]=0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN]=1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

43.3.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																SK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_SK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] is 01b, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^{16} .

43.4 Functional description

43.4.1 Introduction

The SWT is a 32-bit window watchdog timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- a control register (SWT_CR),
- an interrupt register (SWT_IR),
- a time-out register (SWT_TO),
- a window register (SWT_WN),
- a service register (SWT_SR),
- a counter output register (SWT_CO), and
- a service key register (SWT_SK).

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

43.4.1.1 SWT_CR register

The SWT_CR register includes bits to enable the timer, set configuration options, and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific.[1] If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

43.4.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100, in which case the time-out period is set to 0x100. When the SWT is enabled, the time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed. See the Configuration section to determine the source that is used to clock the down counter.

43.4.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

[1] See the chip-specific SWT information.

43.4.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

43.4.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

43.4.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

43.4.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

43.4.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation that is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- fixed service sequence
- keyed service sequence

43.4.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected, which requires writing 0xA602, then 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

43.4.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected, which requires writing two pseudorandom keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudorandom key generator defined by the equation in the following figure. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

$$\mathrm{SK}_{n+1} = (17 \times \mathrm{SK}_n + 3) \bmod 2^{16}$$

Figure 43-1. Pseudorandom Key Generator

43.4.4.3 Window mode

If window mode is enabled (SWT_CR[WND] bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_CR[RIA] bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to 1000, then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

43.4.5 Time-out

The Interrupt Then Reset bit (SWT_CR[ITR]) controls the SWT's action taken when a time-out occurs.

- If the SWT_CR[ITR] bit is 0, the SWT generates a reset request immediately on any time-out.
- If the SWT_CR[ITR] bit is 1, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period.
    - The interrupt is indicated by the Time-out Interrupt Flag (SWT_IR[TIF]).
    - Clear the interrupt by writing 1 to SWT_IR[TIF].

If the service sequence is not written before the second consecutive time-out, the SWT generates a reset request.

43.4.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

Chapter 44
Periodic Interrupt Timer (PIT)

44.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The PIT module is an array of timers that can be used to raise interrupts and trigger DMA channels.

44.1.1 Block diagram

The following figure shows the block diagram of the PIT module.

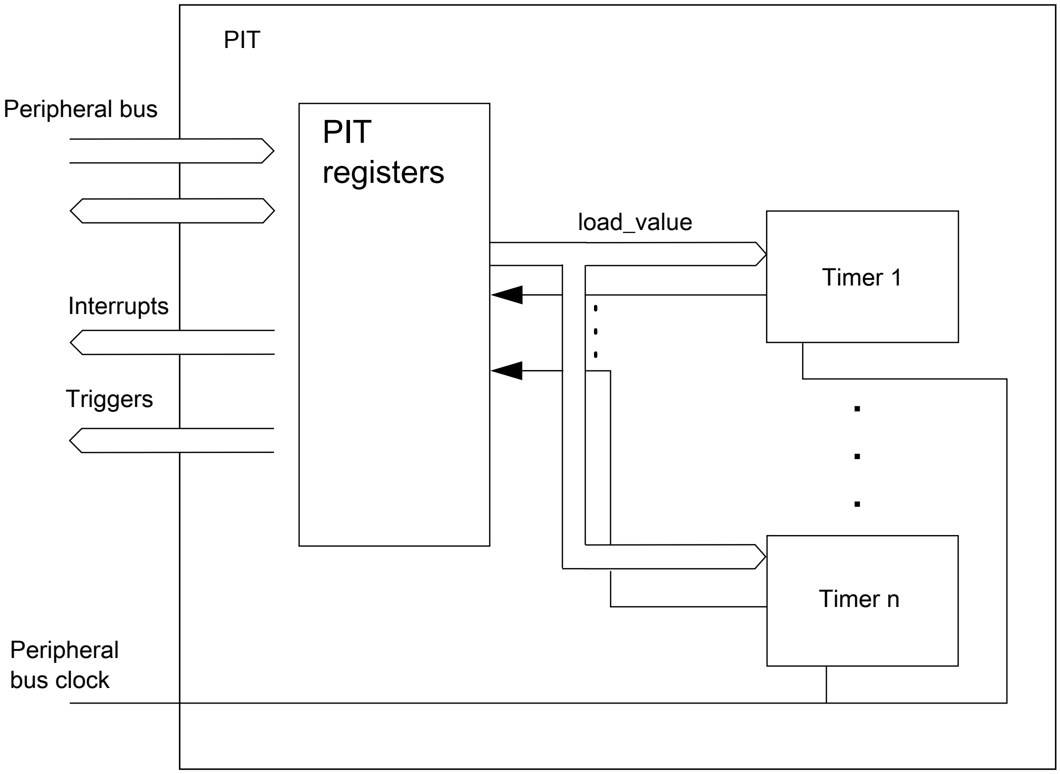


Figure 44-1. Block diagram of the PIT

NOTE

See the chip-specific PIT information for the number of PIT channels used in this MCU.

44.1.2 Features

The main features of this block are:

- Ability of timers to generate DMA trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer

44.2 Signal description

The PIT module has no external pins.

44.3 Memory map/register description

This section provides a detailed description of all registers accessible in the PIT module.

- Reserved registers will read as 0, writes will have no effect.
- See the chip-specific PIT information for the number of PIT channels used in this MCU.

PIT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0000h	44.3.1/1295
E0	PIT Upper Lifetime Timer Register (PIT_LTMR64H)	32	R	0000_0000h	44.3.2/1297
E4	PIT Lower Lifetime Timer Register (PIT_LTMR64L)	32	R	0000_0000h	44.3.3/1297
100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h	44.3.4/1298
104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h	44.3.5/1298
108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h	44.3.6/1299
10C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h	44.3.7/1300
110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h	44.3.4/1298
114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h	44.3.5/1298
118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h	44.3.6/1299
11C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h	44.3.7/1300
120	Timer Load Value Register (PIT_LDVAL2)	32	R/W	0000_0000h	44.3.4/1298
124	Current Timer Value Register (PIT_CVAL2)	32	R	0000_0000h	44.3.5/1298
128	Timer Control Register (PIT_TCTRL2)	32	R/W	0000_0000h	44.3.6/1299
12C	Timer Flag Register (PIT_TFLG2)	32	R/W	0000_0000h	44.3.7/1300
130	Timer Load Value Register (PIT_LDVAL3)	32	R/W	0000_0000h	44.3.4/1298
134	Current Timer Value Register (PIT_CVAL3)	32	R	0000_0000h	44.3.5/1298
138	Timer Control Register (PIT_TCTRL3)	32	R/W	0000_0000h	44.3.6/1299
13C	Timer Flag Register (PIT_TFLG3)	32	R/W	0000_0000h	44.3.7/1300

44.3.1 PIT Module Control Register (PIT_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Access: User read/write

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													Reserved	MDIS	FRZ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_MCR field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved.
30 MDIS	Module Disable - (PIT section) Disables the standard timers. This field must be enabled before any other setup is done. **NOTE:** Always write to this bit after at least 3 bus clock cycles of enabling the PIT clock gate in the device clock generation module. 0 Clock for standard PIT timers is enabled. 1 Clock for standard PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters the Debug mode. 0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

44.3.2 PIT Upper Lifetime Timer Register (PIT_LTMR64H)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

Access: User read only

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTH																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_LTMR64H field descriptions

Field	Description
0–31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

44.3.3 PIT Lower Lifetime Timer Register (PIT_LTMR64L)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Access: User read only

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_LTMR64L field descriptions

Field	Description
0–31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

44.3.4 Timer Load Value Register (PIT_LDVAL*n*)

These registers select the timeout period for the timer interrupts.

Access: User read/write

Address: 0h base + 100h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSV																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_LDVAL*n* field descriptions

Field	Description
0–31 TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

44.3.5 Current Timer Value Register (PIT_CVAL*n*)

These registers indicate the current timer position.

Access: User read only

Address: 0h base + 104h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_CVAL*n* field descriptions

Field	Description
0–31 TVL	Current Timer Value Represents the current timer value, if the timer is enabled.

PIT_CVAL*n* field descriptions (continued)

Field	Description
	NOTE: • If the timer is disabled, do not use this field as its value is unreliable. • The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

44.3.6 Timer Control Register (PIT_TCTRL*n*)

These registers contain the control bits for each timer.

Access: User read/write

Address: 0h base + 108h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TCTRL*n* field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CHN	Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer. For example, for Channel 2, if this field is set, Timer 2 is chained to Timer 1.
30 TIE	Timer Interrupt Enable When an interrupt is pending, or, TFLGn[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first. 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.
31 TEN	Timer Enable Enables or disables the timer. 0 Timer n is disabled. 1 Timer n is enabled.

44.3.7 Timer Flag Register (PIT_TFLG*n*)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TFLG*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

44.4 Functional description

This section provides the functional description of the module.

44.4.1 General operation

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line.

44.4.1.1 Timers

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.

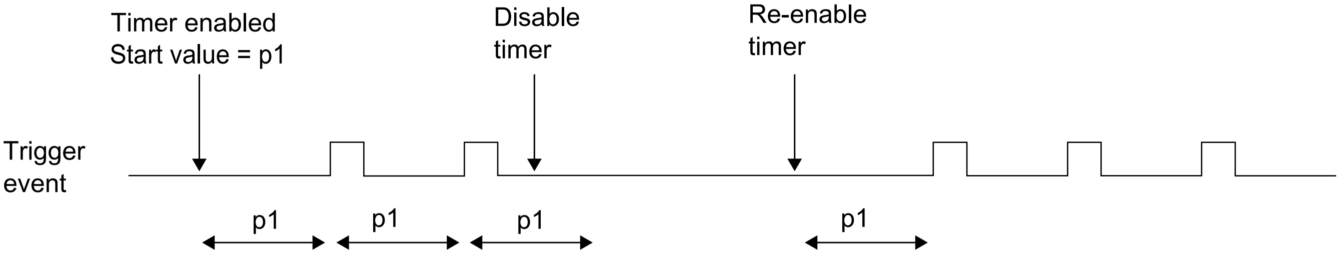


Figure 44-2. Stopping and starting a timer

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.

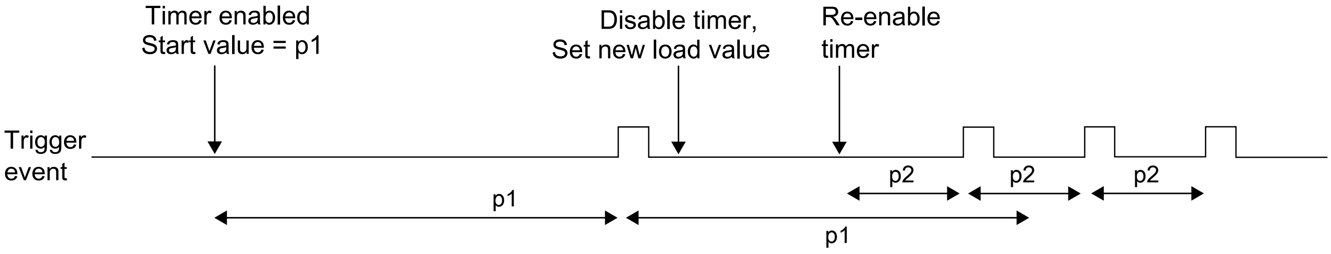


Figure 44-3. Modifying running timer period

It is also possible to change the counter period without restarting the timer by writing LDVAL with the new load value. This value will then be loaded after the next trigger event. See the following figure.

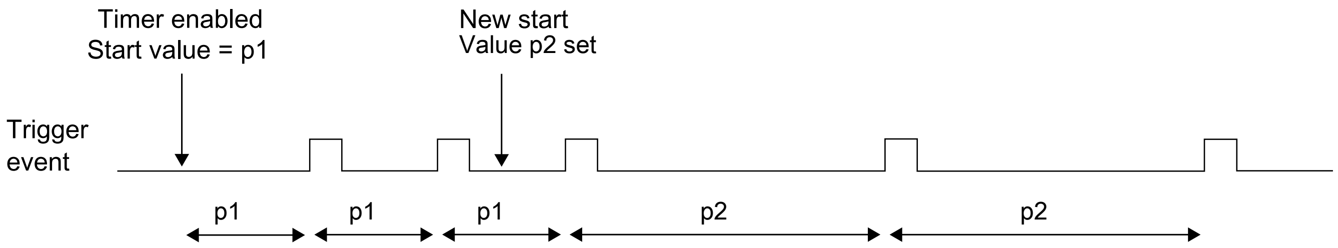


Figure 44-4. Dynamically setting a new load value

44.4.1.2 Debug mode

In Debug mode, the timers will be frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

44.4.2 Interrupts

All the timers support interrupt generation. See the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting TCTRLn[TIE]. TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

44.4.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

44.5 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.

- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns. Timer 1 needs to trigger every 5.12 ms/20 ns = 256,000 cycles and Timer 3 every 30 ms/20 ns = 1,500,000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) -1

This means LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F respectively.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. TCTRL3[TIE] stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1


// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

44.6 Example configuration for chained timers

NOTE

> Check if 100 MHz frequency for PIT timer is valid for your device in the clocking chapter.

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 minute.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN]. TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 2
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2


// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

44.7 Example configuration for the lifetime timer

To configure the lifetimer timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the CTRL register, then the LDVAL registers need to be set to the maximum value.

The timer is a downcounter.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1
```

```
// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

Chapter 45
CAN (FlexCAN)

45.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0 B protocol specification. A general block diagram is shown in the following figure, which describes the main subblocks implemented in the FlexCAN module, including one associated memory for storing message buffers, Receive (Rx) Global Mask registers, Receive Individual Mask registers, Receive FIFO filters, and Receive FIFO ID filters. The functions of the submodules are described in subsequent sections.

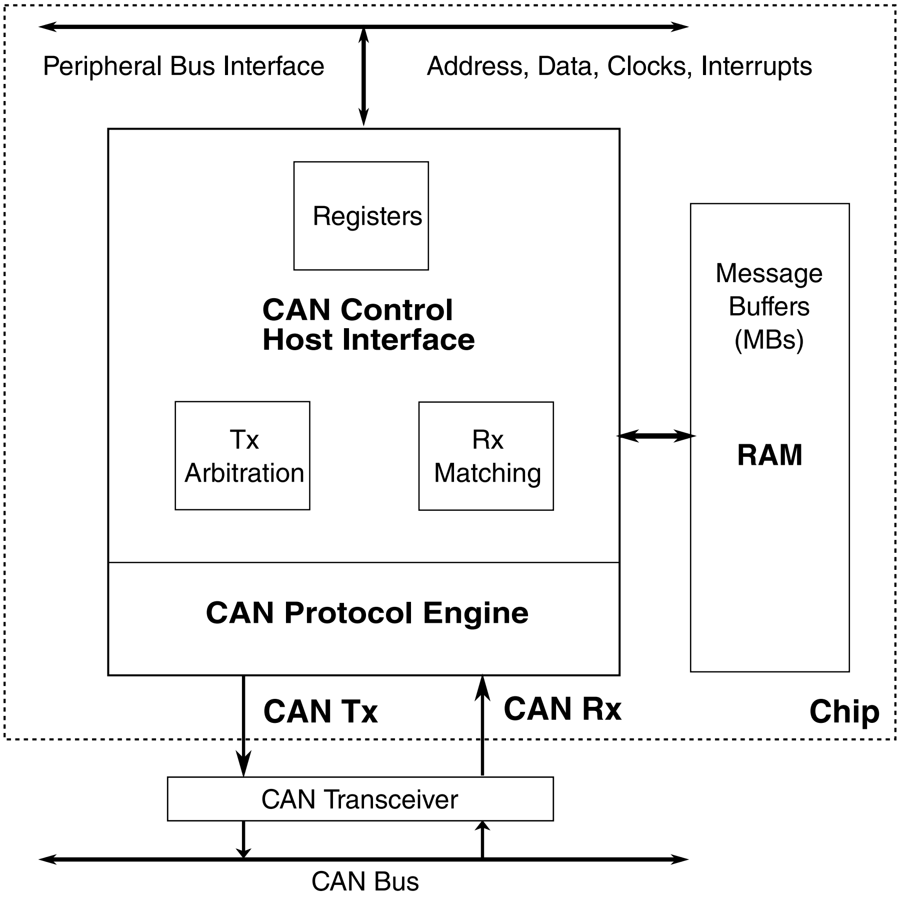


Figure 45-1. FlexCAN block diagram

45.1.1 Overview

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field:

- Real-time processing
- Reliable operation in the EMI environment of a vehicle
- Cost-effectiveness
- Required bandwidth

The FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B, which supports both standard and extended message frames. The message buffers are stored in an embedded RAM dedicated to the FlexCAN module. See the chip-specific FlexCAN information for the actual number of message buffers configured in the chip.

The CAN Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames

- Validating received messages
- Performing error handling

The Controller Host Interface (CHI) sub-module handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms.

The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the BIU.

45.1.2 FlexCAN module features

The FlexCAN module includes these distinctive legacy features:

- Full implementation of the CAN protocol specification, Version 2.0 B
    - Standard data and remote frames
    - Extended data and remote frames
    - Zero to eight bytes data length
    - Programmable bit rate up to 1 Mb/sec
    - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes of zero to eight bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling
- Transmission abort capability
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused structures space can be used as general purpose RAM space
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation

- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes

New major features are also provided:

- Remote request frames may be handled automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous FlexCAN version
- Supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits and the error correction mechanism assures that in this 13-bit word, errors in one bit can be corrected (corrected errors) and errors in 2 bits can be detected but not corrected (non-corrected errors).

45.1.3 Modes of operation

The FlexCAN module has these functional modes:

- Normal mode (User or Supervisor):

  In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze mode:

  Freeze mode is enabled when the FRZ bit in MCR is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when Debug mode is requested at chip level and MCR[FRZ_ACK] is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze mode for more information.

- Listen-Only mode:

  The module enters this mode when the LOM field in the Control 1 Register is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back mode:

  The module enters this mode when the LPB field in the Control 1 Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

For low-power operation, the FlexCAN module has:

- Module Disable mode:

  This low-power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating the MDIS bit in the MCR register. See Module Disable mode for more information.

- Stop mode:

This low power mode is entered when Stop mode is requested at chip level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed. See Stop mode for more information.

45.2 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more detail in the next subsections.

Table 45-1. FlexCAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN Receive Pin	Input
CAN Tx	CAN Transmit Pin	Output

45.2.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

45.2.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

45.3 Memory map/register definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the chip.

45.3.1 FlexCAN memory mapping

The complete memory map for a FlexCAN module is shown in the following table.

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address 0x0080.

Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV field in the MCR register. These registers are identified as S/U in the Access column of Table 45-2.

Table 45-2. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (MCR)	S	Yes	Yes
Control 1 register (CTRL1)	S/U	Yes	No
Free Running Timer register (TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask register (RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask register (RX14MASK)	S/U	No	No
Rx Buffer 15 Mask register (RX15MASK)	S/U	No	No
Error Counter Register (ECR)	S/U	Yes	Yes
Error and Status 1 Register (ESR1)	S/U	Yes	Yes
Interrupt Masks 2 register (IMASK2)	S/U	Yes	Yes
Interrupt Masks 1 register (IMASK1)	S/U	Yes	Yes
Interrupt Flags 2 register (IFLAG2)	S/U	Yes	Yes
Interrupt Flags 1 register (IFLAG1)	S/U	Yes	Yes
Control 2 Register (CTRL2)	S/U	Yes	No
Error and Status 2 Register (ESR2)	S/U	Yes	Yes
CRC Register (CRCR)	S/U	Yes	Yes
Rx FIFO Global Mask register (RXFGMASK)	S/U	No	No
Rx FIFO Information Register (RXFIR)	S/U	No	No
Message buffers	S/U	No	No
Rx Individual Mask Registers	S/U	No	No
Memory Error Control Register (MECR)	S/U	Yes	Yes
Error Injection Address Register (ERRIAR)	S/U	Yes	Yes
Error Injection Data Pattern Register (ERRIDPR)	S/U	Yes	Yes
Error Injection Parity Pattern Register (ERRIPPR)	S/U	Yes	Yes
Error Report Address Register (RERRAR)	S/U	Yes	Yes
Error Report Data Register (RERRDR)	S/U	Yes	Yes
Error Report Syndrome Register (RERRSYNR)	S/U	Yes	Yes
Error Status Register (ERRSR)	S/U	Yes	Yes

The FlexCAN module can store CAN messages for transmission and reception using mailboxes and Rx FIFO structures.

This module's memory map includes sixty-four 128-bit message buffers (MBs) that occupy the range from offset 0x80 to 0x47F .

CAN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (CAN_MCR)	32	R/W	D890_000Fh	45.3.2/1318
4	Control 1 register (CAN_CTRL1)	32	R/W	0000_0000h	45.3.3/1323
8	Free Running Timer (CAN_TIMER)	32	R/W	0000_0000h	45.3.4/1326
10	Rx Mailboxes Global Mask Register (CAN_RXMGMASK)	32	R/W	FFFF_FFFFh	45.3.5/1327
14	Rx 14 Mask register (CAN_RX14MASK)	32	R/W	FFFF_FFFFh	45.3.6/1328
18	Rx 15 Mask register (CAN_RX15MASK)	32	R/W	FFFF_FFFFh	45.3.7/1329
1C	Error Counter (CAN_ECR)	32	R/W	0000_0000h	45.3.8/1329
20	Error and Status 1 register (CAN_ESR1)	32	R/W	0000_0000h	45.3.9/1331
24	Interrupt Masks 2 register (CAN_IMASK2)	32	R/W	0000_0000h	45.3.10/ 1335
28	Interrupt Masks 1 register (CAN_IMASK1)	32	R/W	0000_0000h	45.3.11/ 1335
2C	Interrupt Flags 2 register (CAN_IFLAG2)	32	R/W	0000_0000h	45.3.12/ 1336
30	Interrupt Flags 1 register (CAN_IFLAG1)	32	R/W	0000_0000h	45.3.13/ 1337
34	Control 2 register (CAN_CTRL2)	32	R/W	0080_0000h	45.3.14/ 1339
38	Error and Status 2 register (CAN_ESR2)	32	R/W	0000_0000h	45.3.15/ 1343
44	CRC Register (CAN_CRCR)	32	R	0000_0000h	45.3.16/ 1344
48	Rx FIFO Global Mask register (CAN_RXFGMASK)	32	R/W	FFFF_FFFFh	45.3.17/ 1345
4C	Rx FIFO Information Register (CAN_RXFIR)	32	R	Undefined	45.3.18/ 1346
880	Rx Individual Mask Registers (CAN_RXIMR0)	32	R/W	Undefined	45.3.19/ 1346
884	Rx Individual Mask Registers (CAN_RXIMR1)	32	R/W	Undefined	45.3.19/ 1346
888	Rx Individual Mask Registers (CAN_RXIMR2)	32	R/W	Undefined	45.3.19/ 1346
88C	Rx Individual Mask Registers (CAN_RXIMR3)	32	R/W	Undefined	45.3.19/ 1346
890	Rx Individual Mask Registers (CAN_RXIMR4)	32	R/W	Undefined	45.3.19/ 1346

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
894	Rx Individual Mask Registers (CAN_RXIMR5)	32	R/W	Undefined	45.3.19/ 1346
898	Rx Individual Mask Registers (CAN_RXIMR6)	32	R/W	Undefined	45.3.19/ 1346
89C	Rx Individual Mask Registers (CAN_RXIMR7)	32	R/W	Undefined	45.3.19/ 1346
8A0	Rx Individual Mask Registers (CAN_RXIMR8)	32	R/W	Undefined	45.3.19/ 1346
8A4	Rx Individual Mask Registers (CAN_RXIMR9)	32	R/W	Undefined	45.3.19/ 1346
8A8	Rx Individual Mask Registers (CAN_RXIMR10)	32	R/W	Undefined	45.3.19/ 1346
8AC	Rx Individual Mask Registers (CAN_RXIMR11)	32	R/W	Undefined	45.3.19/ 1346
8B0	Rx Individual Mask Registers (CAN_RXIMR12)	32	R/W	Undefined	45.3.19/ 1346
8B4	Rx Individual Mask Registers (CAN_RXIMR13)	32	R/W	Undefined	45.3.19/ 1346
8B8	Rx Individual Mask Registers (CAN_RXIMR14)	32	R/W	Undefined	45.3.19/ 1346
8BC	Rx Individual Mask Registers (CAN_RXIMR15)	32	R/W	Undefined	45.3.19/ 1346
8C0	Rx Individual Mask Registers (CAN_RXIMR16)	32	R/W	Undefined	45.3.19/ 1346
8C4	Rx Individual Mask Registers (CAN_RXIMR17)	32	R/W	Undefined	45.3.19/ 1346
8C8	Rx Individual Mask Registers (CAN_RXIMR18)	32	R/W	Undefined	45.3.19/ 1346
8CC	Rx Individual Mask Registers (CAN_RXIMR19)	32	R/W	Undefined	45.3.19/ 1346
8D0	Rx Individual Mask Registers (CAN_RXIMR20)	32	R/W	Undefined	45.3.19/ 1346
8D4	Rx Individual Mask Registers (CAN_RXIMR21)	32	R/W	Undefined	45.3.19/ 1346
8D8	Rx Individual Mask Registers (CAN_RXIMR22)	32	R/W	Undefined	45.3.19/ 1346
8DC	Rx Individual Mask Registers (CAN_RXIMR23)	32	R/W	Undefined	45.3.19/ 1346
8E0	Rx Individual Mask Registers (CAN_RXIMR24)	32	R/W	Undefined	45.3.19/ 1346
8E4	Rx Individual Mask Registers (CAN_RXIMR25)	32	R/W	Undefined	45.3.19/ 1346
8E8	Rx Individual Mask Registers (CAN_RXIMR26)	32	R/W	Undefined	45.3.19/ 1346

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8EC	Rx Individual Mask Registers (CAN_RXIMR27)	32	R/W	Undefined	45.3.19/ 1346
8F0	Rx Individual Mask Registers (CAN_RXIMR28)	32	R/W	Undefined	45.3.19/ 1346
8F4	Rx Individual Mask Registers (CAN_RXIMR29)	32	R/W	Undefined	45.3.19/ 1346
8F8	Rx Individual Mask Registers (CAN_RXIMR30)	32	R/W	Undefined	45.3.19/ 1346
8FC	Rx Individual Mask Registers (CAN_RXIMR31)	32	R/W	Undefined	45.3.19/ 1346
900	Rx Individual Mask Registers (CAN_RXIMR32)	32	R/W	Undefined	45.3.19/ 1346
904	Rx Individual Mask Registers (CAN_RXIMR33)	32	R/W	Undefined	45.3.19/ 1346
908	Rx Individual Mask Registers (CAN_RXIMR34)	32	R/W	Undefined	45.3.19/ 1346
90C	Rx Individual Mask Registers (CAN_RXIMR35)	32	R/W	Undefined	45.3.19/ 1346
910	Rx Individual Mask Registers (CAN_RXIMR36)	32	R/W	Undefined	45.3.19/ 1346
914	Rx Individual Mask Registers (CAN_RXIMR37)	32	R/W	Undefined	45.3.19/ 1346
918	Rx Individual Mask Registers (CAN_RXIMR38)	32	R/W	Undefined	45.3.19/ 1346
91C	Rx Individual Mask Registers (CAN_RXIMR39)	32	R/W	Undefined	45.3.19/ 1346
920	Rx Individual Mask Registers (CAN_RXIMR40)	32	R/W	Undefined	45.3.19/ 1346
924	Rx Individual Mask Registers (CAN_RXIMR41)	32	R/W	Undefined	45.3.19/ 1346
928	Rx Individual Mask Registers (CAN_RXIMR42)	32	R/W	Undefined	45.3.19/ 1346
92C	Rx Individual Mask Registers (CAN_RXIMR43)	32	R/W	Undefined	45.3.19/ 1346
930	Rx Individual Mask Registers (CAN_RXIMR44)	32	R/W	Undefined	45.3.19/ 1346
934	Rx Individual Mask Registers (CAN_RXIMR45)	32	R/W	Undefined	45.3.19/ 1346
938	Rx Individual Mask Registers (CAN_RXIMR46)	32	R/W	Undefined	45.3.19/ 1346
93C	Rx Individual Mask Registers (CAN_RXIMR47)	32	R/W	Undefined	45.3.19/ 1346
940	Rx Individual Mask Registers (CAN_RXIMR48)	32	R/W	Undefined	45.3.19/ 1346

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
944	Rx Individual Mask Registers (CAN_RXIMR49)	32	R/W	Undefined	45.3.19/ 1346
948	Rx Individual Mask Registers (CAN_RXIMR50)	32	R/W	Undefined	45.3.19/ 1346
94C	Rx Individual Mask Registers (CAN_RXIMR51)	32	R/W	Undefined	45.3.19/ 1346
950	Rx Individual Mask Registers (CAN_RXIMR52)	32	R/W	Undefined	45.3.19/ 1346
954	Rx Individual Mask Registers (CAN_RXIMR53)	32	R/W	Undefined	45.3.19/ 1346
958	Rx Individual Mask Registers (CAN_RXIMR54)	32	R/W	Undefined	45.3.19/ 1346
95C	Rx Individual Mask Registers (CAN_RXIMR55)	32	R/W	Undefined	45.3.19/ 1346
960	Rx Individual Mask Registers (CAN_RXIMR56)	32	R/W	Undefined	45.3.19/ 1346
964	Rx Individual Mask Registers (CAN_RXIMR57)	32	R/W	Undefined	45.3.19/ 1346
968	Rx Individual Mask Registers (CAN_RXIMR58)	32	R/W	Undefined	45.3.19/ 1346
96C	Rx Individual Mask Registers (CAN_RXIMR59)	32	R/W	Undefined	45.3.19/ 1346
970	Rx Individual Mask Registers (CAN_RXIMR60)	32	R/W	Undefined	45.3.19/ 1346
974	Rx Individual Mask Registers (CAN_RXIMR61)	32	R/W	Undefined	45.3.19/ 1346
978	Rx Individual Mask Registers (CAN_RXIMR62)	32	R/W	Undefined	45.3.19/ 1346
97C	Rx Individual Mask Registers (CAN_RXIMR63)	32	R/W	Undefined	45.3.19/ 1346
AE0	Memory Error Control Register (CAN_MECR)	32	R/W	800C_0080h	45.3.20/ 1347
AE4	Error Injection Address Register (CAN_ERRIAR)	32	R/W	0000_0000h	45.3.21/ 1349
AE8	Error Injection Data Pattern Register (CAN_ERRIDPR)	32	R/W	0000_0000h	45.3.22/ 1350
AEC	Error Injection Parity Pattern Register (CAN_ERRIPPR)	32	R/W	0000_0000h	45.3.23/ 1351
AF0	Error Report Address Register (CAN_RERRAR)	32	R	0000_0000h	45.3.24/ 1351
AF4	Error Report Data Register (CAN_RERRDR)	32	R	0000_0000h	45.3.25/ 1353
AF8	Error Report Syndrome Register (CAN_RERRSYNR)	32	R	0000_0000h	45.3.26/ 1353

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AFC	Error Status Register (CAN_ERRSR)	32	R/W	0000_0000h	45.3.27/ 1356

45.3.2 Module Configuration Register (CAN_MCR)

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDIS	FRZ	RFEN	HALT	NOTRDY	Reserved	SOFTRST	FRZACK	SUPV	Reserved	WRNEN	LPMACK	Reserved	0	SRXDIS	IRMQ
W					▨			▨				▨		▨		
Reset	1	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	LPRIOEN	AEN	0		IDAM		0	MAXMB						
W	▨	▨			▨				▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

CAN_MCR field descriptions

Field	Description
0 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. This is the only bit within this register not affected by soft reset.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	0 Enable the FlexCAN module. 1 Disable the FlexCAN module.
1 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR Register is set or when Debug mode is requested at chip level . When FRZ is asserted, FlexCAN is enabled to enter Freeze mode. Negation of this bit field causes FlexCAN to exit from Freeze mode. 0 Not enabled to enter Freeze mode. 1 Enabled to enter Freeze mode.
2 RFEN	Rx FIFO Enable This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in the table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (in section "Arbitration and Matching Timing"). This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Rx FIFO not enabled. 1 Rx FIFO enabled.
3 HALT	Halt FlexCAN Assertion of this bit puts the FlexCAN module into Freeze mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. Freeze mode cannot be entered while FlexCAN is in a low power mode. 0 No Freeze mode request. 1 Enters Freeze mode if the FRZ bit is asserted.
4 NOTRDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable mode , Stop mode or Freeze mode. It is negated once FlexCAN has exited these modes. 0 FlexCAN module is either in Normal mode, Listen-Only mode or Loop-Back mode. 1 FlexCAN module is either in Disable mode , Stop mode or Freeze mode.
5 Reserved	This field is reserved. Always write 0 to this field.
6 SOFTRST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER , ECR, ESR1, ESR2, IMASK1, IMASK2, IFLAG1, IFLAG2 and CRCR. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: CTRL1, CTRL2, all RXIMR registers, RXMGMASK, RX14MASK, RX15MASK, RXFGMASK, RXFIR, all Message Buffers . The SOFTRST bit can be asserted directly by the CPU when it writes to the MCR Register, but it is also asserted when global soft reset is requested at chip level . Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	Soft reset cannot be applied while clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied. 0 No reset request. 1 Resets the registers affected by soft reset.
7 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZACK bit to know when FlexCAN has actually entered Freeze mode. If Freeze Mode request is negated, then this bit is negated after the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Section "Freeze Mode". **NOTE:** FRZACK will be asserted within 178 CAN bits from the freeze mode request by the CPU, and negated within 2 CAN bits after the freeze mode request removal (see Section "Protocol Timing"). 0 FlexCAN not in Freeze mode, prescaler running. 1 FlexCAN in Freeze mode, prescaler stopped.
8 SUPV	Supervisor Mode This bit configures the FlexCAN to be either in Supervisor or User mode. The registers affected by this bit are marked as S/U in the Access Type column of the module memory map. Reset value of this bit is 1, so the affected registers start with Supervisor access allowance only . This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 FlexCAN is in User mode. Affected registers allow both Supervisor and Unrestricted accesses . 1 FlexCAN is in Supervisor mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location .
9 Reserved	This field is reserved. Always write 0 to this field.
10 WRNEN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register. If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 TWRNINT and RWRNINT bits are zero, independent of the values in the error counters. 1 TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96.
11 LPMACK	Low-Power Mode Acknowledge This read-only bit indicates that FlexCAN is in a low-power mode (Disable mode , Stop mode). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. **NOTE:** LPMACK will be asserted within 180 CAN bits from the low-power mode request by the CPU, and negated within 2 CAN bits after the low-power mode request removal (see Section "Protocol Timing"). 0 FlexCAN is not in a low-power mode. 1 FlexCAN is in a low-power mode.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
12 Reserved	This field is reserved. Always write 0 to this field.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SRXDIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Self reception enabled. 1 Self reception disabled.
15 IRMQ	Individual Rx Masking And Queue Enable This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK and RX15MASK, RXFGMASK. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY. 1 Individual Rx masking and queue feature are enabled.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 LPRIOEN	Local Priority Enable This bit is provided for backwards compatibility with legacy applications. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Local Priority disabled. 1 Local Priority enabled.
19 AEN	Abort Enable This bit is supplied for backwards compatibility with legacy applications. When asserted, it enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** When MCR[AEN] is asserted, only the abort mechanism (see Section "Transmission Abort Mechanism") must be used for updating Mailboxes configured for transmission. **CAUTION:** Writing the Abort code into Rx Mailboxes can cause unpredictable results when the MCR[AEN] is asserted. 0 Abort disabled. 1 Abort enabled.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 IDAM	ID Acceptance Mode This 2-bit field identifies the format of the Rx FIFO ID Filter Table elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section "Rx FIFO Structure". This field can be written only in Freeze mode because it is blocked by hardware in other modes. 00 Format A: One full ID (standard and extended) per ID Filter Table element. 01 Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element. 10 Format C: Four partial 8-bit Standard IDs per ID Filter Table element. 11 Format D: All frames rejected.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 MAXMB	Number Of The Last Message Buffer This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB **NOTE:** MAXMB must be programmed with a value smaller than the parameter NUMBER_OF_MB, otherwise the number of the last effective Message Buffer will be: (NUMBER_OF_MB - 1) Additionally, the value of MAXMB must encompass the FIFO size defined by CTRL2[RFFN]. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (in Section "Arbitration and Matching Timing").

45.3.3 Control 1 register (CAN_CTRL1)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back mode, Listen-Only mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BOFFMSK	ERRMSK	CLKSRC	LPB	TWRNMSK	RWRNMSK	0		SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
W							▨									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CTRL1 field descriptions

Field	Description
0–7 PRESDIV	Prescaler Division Factor This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this field is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Section "Protocol Timing". This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRESDIV + 1)
8–9 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. One time quantum is equal to the Sclock period. The valid programmable values are 0–3. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = RJW + 1.
10–12 PSEG1	Phase Segment 1 This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta.
13–15 PSEG2	Phase Segment 2

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta.
16 BOFFMSK	Bus Off Mask This bit provides a mask for the Bus Off Interrupt. 0 Bus Off interrupt disabled. 1 Bus Off interrupt enabled.
17 ERRMSK	Error Mask This bit provides a mask for the Error Interrupt. 0 Error interrupt disabled. 1 Error interrupt enabled.
18 CLKSRC	CAN Engine Clock Source This bit selects the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock). In order to guarantee reliable operation, this bit can be written only in Disable mode because it is blocked by hardware in other modes. See Section "Protocol Timing". 0 The CAN engine clock source is the oscillator clock. Under this condition, the oscillator clock frequency must be lower than the bus clock. 1 The CAN engine clock source is the peripheral clock.
19 LPB	Loop Back Mode This bit configures FlexCAN to operate in Loop-Back mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** In this mode, the MCR[SRXDIS] cannot be asserted because this will impede the self reception of a transmitted message. 0 Loop Back disabled. 1 Loop Back enabled.
20 TWRNMSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRNINT flag in the Error and Status Register. This bit is read as zero when MCR[WRNEN] bit is negated. This bit can be written only if MCR[WRNEN] bit is asserted. 0 Tx Warning Interrupt disabled. 1 Tx Warning Interrupt enabled.
21 RWRNMSK	Rx Warning Interrupt Mask

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	This bit provides a mask for the Rx Warning Interrupt associated with the RWRNINT flag in the Error and Status Register. This bit is read as zero when MCR[WRNEN] bit is negated. This bit can be written only if MCR[WRNEN] bit is asserted. 0 Rx Warning Interrupt disabled. 1 Rx Warning Interrupt enabled.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 SMP	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Just one sample is used to determine the bit value. 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples; a majority rule is used.
25 BOFFREC	Bus Off Recovery This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during Bus Off, but it will be effective only the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B. 1 Automatic recovering from Bus Off state disabled.
26 TSYN	Timer Sync This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message, that is, global network time. If the RFEN bit in MCR is set (Rx FIFO enabled), the first available Mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Timer Sync feature disabled 1 Timer Sync feature enabled
27 LBUF	Lowest Buffer Transmitted First This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIOEN bit does not affect the priority arbitration. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Buffer with highest priority is transmitted first. 1 Lowest number buffer is transmitted first.
28 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the REC, as if it was trying to acknowledge the message. Listen-Only mode acknowledgement can be obtained by the state of ESR1[FLTCONF] field which is Passive Error when Listen-Only mode is entered. There can be some delay between the Listen-Only mode request and acknowledge. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Listen-Only mode is deactivated. 1 FlexCAN module operates in Listen-Only mode.
29–31 PROPSEG	Propagation Segment This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.

45.3.4 Free Running Timer (CAN_TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable , Stop, and Freeze modes.

The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CTRL1[TSYN] is asserted, the Timer is reset whenever a message is received in the first available Mailbox, according to CTRL2[RFFN] setting.

The CPU can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure; see Section "Mailbox Lock Mechanism".

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TIMER															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_TIMER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TIMER	Timer Value Contains the free-running counter value.

45.3.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK)

This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When the MCR[IRMQ] bit is negated, RXMGMASK is always in effect.
- When the MCR[IRMQ] bit is asserted, RXMGMASK has no effect.

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MG[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CAN_RXMGMASK field descriptions

Field	Description
0–31 MG[31:0]	Rx Mailboxes Global Mask Bits These bits mask the Mailbox filter bits. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the Mailbox. The following table shows in detail which MG bits mask each Mailbox filter field.

CAN_RXMGMASK field descriptions (continued)

Field	Description

SMB[RTR] [1]	CTRL2[RRS]	CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	Reserved
0	-	0	note [2]	note [3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

[1] RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
[2] If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
[3] If the CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

45.3.6 Rx 14 Mask register (CAN_RX14MASK)

This register is located in RAM.

RX14MASK is provided for legacy application support. When the MCR[IRMQ] bit is asserted, RX14MASK has no effect.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RX14M[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CAN_RX14MASK field descriptions

Field	Description
0–31 RX14M[31:0]	Rx Buffer 14 Mask Bits Each mask bit masks the corresponding Mailbox 14 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register.

CAN_RX14MASK field descriptions (continued)

Field	Description
	0 The corresponding bit in the filter is "don’t care." 1 The corresponding bit in the filter is checked.

45.3.7 Rx 15 Mask register (CAN_RX15MASK)

This register is located in RAM.

RX15MASK is provided for legacy application support. When the MCR[IRMQ] bit is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can be programmed only while the module is in Freeze mode because it is blocked by hardware in other modes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RX15M[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CAN_RX15MASK field descriptions

Field	Description
0–31 RX15M[31:0]	Rx Buffer 15 Mask Bits Each mask bit masks the corresponding Mailbox 15 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don’t care." 1 The corresponding bit in the filter is checked.

45.3.8 Error Counter (CAN_ECR)

This register has two 8-bit fields reflecting the value of two FlexCAN error counters: Transmit Error Counter (TXERRCNT field) and Receive Error Counter (RXERRCNT field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read-only except in Freeze mode, where they can be written by the CPU.

FlexCAN responds to any bus state as described in the protocol, for example, transmit Error Active or Error Passive flag, delay its transmission start time (Error Passive) and avoid any influence on the bus when in Bus Off state.

The following are the basic rules for FlexCAN bus state transitions:

- If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the FLTCONF field in the Error and Status Register is updated to reflect ‘Error Passive’ state.
- If the FlexCAN state is ‘Error Passive’, and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLTCONF field in the Error and Status Register is updated to reflect ‘Error Active’ state.
- If the value of TXERRCNT increases to be greater than 255, the FLTCONF field in the Error and Status Register is updated to reflect ‘Bus Off’ state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in ‘Bus Off’ state, then TXERRCNT is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, the FLTCONF field in the Error and Status Register is updated to be ‘Error Active’ and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value.
- If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in the Error and Status Register). After the transition to ‘Error Passive’ state, the TXERRCNT does not increment anymore by acknowledge errors. Therefore the device never goes to the ‘Bus Off’ state.
- If the RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to ‘Error Active’ state.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RXERRCNT								TXERRCNT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ECR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 RXERRCNT	Receive Error Counter

Table continues on the next page...

CAN_ECR field descriptions (continued)

Field	Description
24–31 TXERRCNT	Transmit Error Counter

45.3.9 Error and Status 1 register (CAN_ESR1)

This register reflects various error conditions and some general status of the device. It is also the source of interrupts to the CPU.

A CPU read operation clears the following fields to 0, so these fields report error conditions that occurred after the last time the CPU read this register: BIT1ERR, BIT0ERR, ACKERR, CRCERR, FRMERR, and STFERR. TXWRN, RXWRN, IDLE, TX, FLTCONF, and RX provide status information.

The following table shows the FlexCAN state variables and their meanings. Combinations not shown in the table are reserved.

SYNCH	IDLE	TX	RX	FlexCAN state
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													SYNCH	TWRNINT	RWRNINT
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR1 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SYNCH	CAN Synchronization Status This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not synchronized to the CAN bus. 1 FlexCAN is synchronized to the CAN bus.
14 TWRNINT	Tx Warning Interrupt Flag If the WRNEN bit in MCR is asserted, the TWRNINT bit is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRNMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This flag is not generated during Bus Off state. This bit is not updated during Freeze mode. 0 No such occurrence. 1 The Tx error counter transitioned from less than 96 to greater than or equal to 96.
15 RWRNINT	Rx Warning Interrupt Flag If the WRNEN bit in MCR is asserted, the RWRNINT bit is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRNMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This bit is not updated during Freeze mode. 0 No such occurrence. 1 The Rx error counter transitioned from less than 96 to greater than or equal to 96.
16 BIT1ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	NOTE: This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 0 No such occurrence. 1 At least one bit sent as recessive is received as dominant.
17 BIT0ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence. 1 At least one bit sent as dominant is received as recessive.
18 ACKERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT. 0 No such occurrence. 1 An ACK error occurred since last read of this register.
19 CRCERR	Cyclic Redundancy Check Error This bit indicates that a CRC Error has been detected by the receiver node, that is, the calculated CRC is different from the received. 0 No such occurrence. 1 A CRC error occurred since last read of this register.
20 FRMERR	Form Error This bit indicates that a Form Error has been detected by the receiver node, that is, a fixed-form bit field contains at least one illegal bit. 0 No such occurrence. 1 A Form Error occurred since last read of this register.
21 STFERR	Stuffing Error This bit indicates that a Stuffing Error has been etected. 0 No such occurrence. 1 A Stuffing Error occurred since last read of this register.
22 TXWRN	TX Error Warning This bit indicates when repetitive errors are occurring during message transmission. This bit is not updated during Freeze mode. 0 No such occurrence. 1 TXERRCNT is greater than or equal to 96.
23 RXWRN	Rx Error Warning This bit indicates when repetitive errors are occurring during message reception. This bit is not updated during Freeze mode. 0 No such occurrence. 1 RXERRCNT is greater than or equal to 96.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
24 IDLE	This bit indicates when CAN bus is in IDLE state. See the table in the overall CAN_ESR1 register description. 0 No such occurrence. 1 CAN bus is now IDLE.
25 TX	FlexCAN In Transmission This bit indicates if FlexCAN is transmitting a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not transmitting a message. 1 FlexCAN is transmitting a message.
26–27 FLTCONF	Fault Confinement State This 2-bit field indicates the Confinement State of the FlexCAN module. If the LOM bit in the Control Register is asserted, after some delay that depends on the CAN bit timing the FLTCONF field will indicate “Error Passive”. The very same delay affects the way how FLTCONF reflects an update to ECR register by the CPU. It may be necessary up to one CAN bit time to get them coherent again. Because the Control Register is not affected by soft reset, the FLTCONF field will not be affected by soft reset if the LOM bit is asserted. 00 Error Active 01 Error Passive 1x Bus Off
28 RX	FlexCAN In Reception This bit indicates if FlexCAN is receiving a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not receiving a message. 1 FlexCAN is receiving a message.
29 BOFFINT	Bus Off Interrupt This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register (BOFFMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 FlexCAN module entered Bus Off state.
30 ERRINT	Error Interrupt This bit indicates that at least one of the Error Bits (bits 15-10) is set. If the corresponding mask bit CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 Indicates setting of any Error Bit in the Error and Status Register.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.3.10 Interrupt Masks 2 register (CAN_IMASK2)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB63 to MB32. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding IFLAG2 bit is set.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUFHM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK2 field descriptions

Field	Description
0–31 BUFHM	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB63 to MB32. **NOTE:** Setting or clearing a bit in the IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

45.3.11 Interrupt Masks 1 register (CAN_IMASK1)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB31 to MB0. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding IFLAG1 bit is set.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUFLM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK1 field descriptions

Field	Description
0–31 BUFLM	Buffer MB $_i$ Mask

CAN_IMASK1 field descriptions (continued)

Field	Description
	Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB0. **NOTE:** Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

45.3.12 Interrupt Flags 2 register (CAN_IFLAG2)

This register defines the flags for the 32 Message Buffer interrupts for MB63 to MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating MCR[MAXMB] field, CPU must service the IFLAG2 bits whose MB value is greater than the MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUFHI																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG2 field descriptions

Field	Description
0–31 BUFHI	Buffer MB $_i$ Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB63 to MB32. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.

45.3.13 Interrupt Flags 1 register (CAN_IFLAG1)

This register defines the flags for the 32 Message Buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the bit MCR[RFEN] is set, the function of the 8 least significant interrupt flags BUF[7:0]I changes: BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, and the BUF4TO0I field is reserved.

Before enabling the RFEN, the CPU must service the IFLAG bits asserted in the Rx FIFO region; see Section "Rx FIFO". Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the RFEN bit is negated, the FIFO flags must be cleared. The same care must be taken when an RFFN value is selected extending Rx FIFO filters beyond MB7. For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

Before updating MCR[MAXMB] field, CPU must service the IFLAG1 bits whose MB value is greater than the MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUF31TO8I															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF31TO8I								BUF7I	BUF6I	BUF5I	BUF4TO1I				BUF0I
W	w1c								w1c	w1c	w1c	w1c				w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG1 field descriptions

Field	Description
0–23 BUF31TO8I	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB31 to MB8. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.
24 BUF7I	Buffer MB7 Interrupt Or "Rx FIFO Overflow" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF7I flag represents "Rx FIFO Overflow" when MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. 0 No occurrence of MB7 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO overflow when MCR[RFEN]=1 1 MB7 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO overflow when MCR[RFEN]=1
25 BUF6I	Buffer MB6 Interrupt Or "Rx FIFO Warning" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF6I flag represents "Rx FIFO Warning" when MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than 4. 0 No occurrence of MB6 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO almost full when MCR[RFEN]=1 1 MB6 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO almost full when MCR[RFEN]=1
26 BUF5I	Buffer MB5 Interrupt Or "Frames available in Rx FIFO" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB5. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF5I flag represents "Frames available in Rx FIFO" when MCR[RFEN] is set. In this case, the flag indicates that at least one frame is available to be read from the Rx FIFO. 0 No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the FIFO, when MCR[RFEN]=1 1 MB5 completed transmission/reception when MCR[RFEN]=0, or frame(s) available in the Rx FIFO when MCR[RFEN]=1
27–30 BUF4TO1I	Buffer MB $_i$ Interrupt Or "reserved" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1. **NOTE:** These flags are cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF4TO1I flags are reserved when MCR[RFEN] is set.

Table continues on the next page...

CAN_IFLAG1 field descriptions (continued)

Field	Description
	0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.
31 BUF0I	Buffer MB0 Interrupt Or "reserved" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB0. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF0I flag is reserved when MCR[RFEN] is set. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.

45.3.14 Control 2 register (CAN_CTRL2)

This register contains control bits for CAN errors, FIFO features, and mode selection.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		ECRWRE	WRMFRZ	RFFN				TASD					MRP	RRS	EACEN
W	▨	▨														
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CTRL2 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 ECRWRE	Error-correction Configuration Register Write Enable Enable that the MECR register is updated. This bit is automatically set to 0 if the protocol described in the "Detection and Correction of Memory Errors" section is not followed. 0 Disable update. 1 Enable update.

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>3
WRMFRZ</td><td>Write-Access To Memory In Freeze Mode

Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode.

0 Maintain the write access restrictions.
1 Enable unrestricted write access to FlexCAN memory.</td></tr>
<tr><td>4–7
RFFN</td><td>Number Of Rx FIFO Filters

This 4-bit field defines the number of Rx FIFO filters, as shown in the following table. The maximum selectable number of filters is determined by the chip. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that make the number of Message Buffers occupied by Rx FIFO and ID Filter exceed the number of Mailboxes present, defined by MCR[MAXMB].

<b>NOTE:</b> Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available.

Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value correponding to a number of filters not greater than the number of available memory words which can be calculated as follows:

(SETUP_MB - 6) × 4

where SETUP_MB is the least between NUMBER_OF_MB and MAXMB.

The number of remaining Mailboxes available will be:

(SETUP_MB - 8) - (RFFN × 2)

If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value (memory space available) the exceeding ones will not be functional.

<table>
<tr><th>RFFN[3:0]</th><th>Number of Rx FIFO filters</th><th>Message Buffers occupied by Rx FIFO and ID Filter Table</th><th>Remaining Available Mailboxes[1]</th><th>Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks</th><th>Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask [2]</th></tr>
<tr><td>0x0</td><td>8</td><td>MB 0-7</td><td>MB 8-63</td><td>Elements 0-7</td><td>none</td></tr>
<tr><td>0x1</td><td>16</td><td>MB 0-9</td><td>MB 10-63</td><td>Elements 0-9</td><td>Elements 10-15</td></tr>
<tr><td>0x2</td><td>24</td><td>MB 0-11</td><td>MB 12-63</td><td>Elements 0-11</td><td>Elements 12-23</td></tr>
<tr><td>0x3</td><td>32</td><td>MB 0-13</td><td>MB 14-63</td><td>Elements 0-13</td><td>Elements 14-31</td></tr>
<tr><td>0x4</td><td>40</td><td>MB 0-15</td><td>MB 16-63</td><td>Elements 0-15</td><td>Elements 16-39</td></tr>
<tr><td>0x5</td><td>48</td><td>MB 0-17</td><td>MB 18-63</td><td>Elements 0-17</td><td>Elements 18-47</td></tr>
<tr><td>0x6</td><td>56</td><td>MB 0-19</td><td>MB 20-63</td><td>Elements 0-19</td><td>Elements 20-55</td></tr>
<tr><td>0x7</td><td>64</td><td>MB 0-21</td><td>MB 22-63</td><td>Elements 0-21</td><td>Elements 22-63</td></tr>
<tr><td>0x8</td><td>72</td><td>MB 0-23</td><td>MB 24-63</td><td>Elements 0-23</td><td>Elements 24-71</td></tr>
<tr><td>0x9</td><td>80</td><td>MB 0-25</td><td>MB 26-63</td><td>Elements 0-25</td><td>Elements 26-79</td></tr>
<tr><td>0xA</td><td>88</td><td>MB 0-27</td><td>MB 28-63</td><td>Elements 0-27</td><td>Elements 28-87</td></tr>
<tr><td>0xB</td><td>96</td><td>MB 0-29</td><td>MB 30-63</td><td>Elements 0-29</td><td>Elements 30-95</td></tr>
<tr><td>0xC</td><td>104</td><td>MB 0-31</td><td>MB 32-63</td><td>Elements 0-31</td><td>Elements 32-103</td></tr>
<tr><td>0xD</td><td>112</td><td>MB 0-33</td><td>MB 34-63</td><td>Elements 0-31</td><td>Elements 32-111</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

RFFN[3:0]	Number of Rx FIFO filters	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes[1]	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask [2]
0xE	120	MB 0-35	MB 36-63	Elements 0-31	Elements 32-119
0xF	128	MB 0-37	MB 38-63	Elements 0-31	Elements 32-127

1. The number of the last remaining available mailboxes is defined by the least value between the parameter NUMBER_OF_MB minus 1 and the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

Field	Description
8–12 TASD	Tx Arbitration Start Delay This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. This field can be written only in Freeze mode because it is blocked by hardware in other modes. This field is useful to optimize the transmit performance based on factors such as: peripheral/serial clock ratio, CAN bit timing and number of MBs. The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and CAN baud rate and inversely proportional to the peripheral clock frequency. The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. Therefore, if there are few MBs and the system/serial clock ratio is high and the CAN baud rate is low then the arbitration can be delayed and vice-versa. If TASD is 0 then the arbitration start is not delayed, thus the CPU has less time to configure a Tx MB for the next arbitration, but more time is reserved for arbitration. On the other hand, if TASD is 24 then the CPU can configure a Tx MB later and less time is reserved for arbitration. If too little time is reserved for arbitration the FlexCAN may be not able to find winner MBs in time to compete with other nodes for the CAN bus. If the arbitration ends too much time before the first bit of Intermission field then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is not the best to be transmitted. The optimal configuration for TASD can be calculated as: $TASD = 25 - \{f_{CANCLK} \times [MAXMB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)] \times 2\} / \{f_{SYS} \times [1+(PSEG1+1)+(PSEG2+1)+(PROPSEG+1)] \times (PRESDIV+1)\}$ where: • f_{CANCLK} is the Protocol Engine (PE) Clock (see section "Protocol Timing"), in Hz • f_{SYS} is the peripheral clock, in Hz • MAXMB is the value in CTRL1[MAXMB] field • RFEN is the value in CTRL1[RFEN] bit • RFFN is the value in CTRL2[RFFN] field • PSEG1 is the value in CTRL1[PSEG1] field • PSEG2 is the value in CTRL1[PSEG2] field • PROPSEG is the value in CTRL1[PROPSEG] field • PRESDIV is the value in CTRL1[PRESDIV] field

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	See Section "Arbitration process" and Section "Protocol Timing" for more details.
13 MRP	Mailboxes Reception Priority If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Matching starts from Rx FIFO and continues on Mailboxes. 1 Matching starts from Mailboxes and continues on Rx FIFO.
14 RRS	Remote Request Storing If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated. If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Remote Response Frame is generated. 1 Remote Request Frame is stored.
15 EACEN	Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes This bit controls the comparison of IDE and RTR bits whithin Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Rx Mailbox filter’s IDE bit is always compared and RTR is never compared despite mask bits. 1 Enables the comparison of both Rx Mailbox filter’s IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

1. The number of the last remaining available mailboxes is defined by the least value between the parameter NUMBER_OF_MB minus 1 and the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

45.3.15 Error and Status 2 register (CAN_ESR2)

This register reflects various interrupt flags and some general status.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0									LPTM						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VPS	IMB	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR2 field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LPTM	Lowest Priority Tx Mailbox If ESR2[VPS] is asserted, this field indicates the lowest number inactive Mailbox (see the IMB bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CTRL1[LBUF] bit value. If CTRL1[LBUF] bit is negated then the Mailbox indicated is the one that has the greatest arbitration value (see the "Highest priority Mailbox first" section). If CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VPS	Valid Priority Status This bit indicates whether IMB and LPTM contents are currently valid or not. VPS is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a Mailbox that has already been scanned, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox that is being transmitted then VPS is not asserted. VPS is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox. **NOTE:** ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of a MB that is blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of a MB that is being transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked. 0 Contents of IMB and LPTM are invalid. 1 Contents of IMB and LPTM are valid.
18 IMB	Inactive Mailbox If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases:

Table continues on the next page...

CAN_ESR2 field descriptions (continued)

Field	Description
	• During arbitration, if an LPTM is found and it is inactive. • If IMB is not asserted and a frame is transmitted successfully. This bit is cleared in all start of arbitration (see Section "Arbitration process"). **NOTE:** LPTM mechanism have the following behavior: if an MB is successfully transmitted and ESR2[IMB]=0 (no inactive Mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM]. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one.
19–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.3.16 CRC Register (CAN_CRCR)

This register provides information about the CRC of transmitted messages.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0									MBCRC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TXCRC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CRCR field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBCRC	CRC Mailbox This field indicates the number of the Mailbox corresponding to the value in TXCRC field.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 TXCRC	CRC Transmitted This field indicates the CRC value of the last message transmitted. This field is updated at the same time the Tx Interrupt Flag is asserted.

45.3.17 Rx FIFO Global Mask register (CAN_RXFGMASK)

This register is located in RAM.

If Rx FIFO is enabled RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FGM[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CAN_RXFGMASK field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–31
FGM[31:0]</td><td>Rx FIFO Global Mask Bits
These bits mask the ID Filter Table elements bits in a perfect alignment.
The following table shows how the FGM bits correspond to each IDAF field.
<table>
<tr><th rowspan="2">Rx FIFO ID Filter Table Elements Format (MCR[IDAM])</th><th colspan="6">Identifier Acceptance Filter Fields</th></tr>
<tr><th>RTR</th><th>IDE</th><th>RXIDA</th><th>RXIDB [1]</th><th>RXIDC [2]</th><th>Reserved</th></tr>
<tr><td>A</td><td>FGM[31]</td><td>FGM[30]</td><td>FGM[29:1]</td><td>-</td><td rowspan="2">-</td><td>FGM[0]</td></tr>
<tr><td>B</td><td>FGM[31],
FGM[15]</td><td>FGM[30],
FGM[14]</td><td rowspan="2">-</td><td>FGM[29:16],
FGM[13:0]</td><td rowspan="2">-</td></tr>
<tr><td>C</td><td>-</td><td>-</td><td>-</td><td>FGM[31:24],
FGM[23:16],
FGM[15:8],
FGM[7:0]</td></tr>
</table>
1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
0 The corresponding bit in the filter is "don’t care."
1 The corresponding bit in the filter is checked.</td></tr>
</table>

[1] If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

[2] If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

45.3.18 Rx FIFO Information Register (CAN_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. See Section "Rx FIFO" for instructions on reading this register.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							IDHIT								
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXFIR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 IDHIT	Identifier Acceptance Filter Hit Indicator This field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the IFLAG[BUF5I] is asserted.

45.3.19 Rx Individual Mask Registers (CAN_RXIMR*n*)

These registers are located in RAM.

RXIMR are used as acceptance masks for ID filtering in Rx MBs and the Rx FIFO. If the Rx FIFO is not enabled, one mask register is provided for each available Mailbox, providing ID masking capability on a per Mailbox basis.

When the Rx FIFO is enabled (MCR[RFEN] bit is asserted), up to 32 Rx Individual Mask Registers can apply to the Rx FIFO ID Filter Table elements on a one-to-one correspondence depending on the setting of CTRL2[RFFN].

RXIMR can only be written by the CPU while the module is in Freeze mode; otherwise, they are blocked by hardware.

The Individual Rx Mask Registers are not affected by reset and must be explicitly initialized prior to any reception.

Address: 0h base + 880h offset + (4d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MI[31:0]																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXIMR*n* field descriptions

Field	Description
0–31 MI[31:0]	Individual Mask Bits Each Individual Mask Bit masks the corresponding bit in both the Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filters, see the RXMGMASK register description. For Rx FIFO ID Filter Table elements, see the RXFGMASK register description. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

45.3.20 Memory Error Control Register (CAN_MECR)

This register contains control bits for memory error detection and correction (ECC).

NOTE

> When bit CTRL2[ECRWRE] is 0, this register is read-only and writes to this register result in bus transfer errors.

Address: 0h base + AE0h offset = AE0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ECRWRDIS	0														CEI_MSK
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HAERRIE	FAERRIE	EXTERRIE	0			RERRDIS	ECCDIS	NCEFAFRZ	0						
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

CAN_MECR field descriptions

Field	Description
0 ECRWRDIS	Error Configuration Register Write Disable Disables writes on this register. 0 Write is enabled. 1 Write is disabled.
1–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CEI_MSK	Correctable Errors Interrupt Mask Enables the interrupt in case of correctable errors detected in memory reads issued by the host or FlexCAN internal processes. 0 Interrupt is disabled. 1 Interrupt is enabled.
16 HAERRIE	Host Access Error Injection Enable Enables the injection of errors only in memory reads issued by the host (CPU). 0 Injection is disabled. 1 Injection is enabled.
17 FAERRIE	FlexCAN Access Error Injection Enable Enables the injection of errors only in memory reads issued by the FlexCAN internal processes. 0 Injection is disabled. 1 Injection is enabled.
18 EXTERRIE	Extended Error Injection Enable Memory accesses performed by internal FlexCAN processes are 64-bit. This bit extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data and parity words. See Error Injection Data Pattern Register (FCERRIDPR) and Error Injection Parity Pattern Register (FCERRIPPR) 0 Error injection is applied only to the 32-bit word. 1 Error injection is applied to the 64-bit word.
19–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RERRDIS	Error Report Disable Disables the update of the error report registers. The update of error-related flags and the generation of bus transfer errors are still active.

Table continues on the next page...

CAN_MECR field descriptions (continued)

Field	Description
	NOTE: When reading the report registers, this bit must be cleared to assure coherence on the consecutive register reads. 0 Enable updates of the error report registers. 1 Disable updates of the error report registers.
23 ECCDIS	Error Correction Disable Disables completely the memory error detection and correction mechanism. Besides disabling the error report mechanism, it also stops the update of the error-related flags and generation of bus transfer errors. The parity bits continue being calculated and written into memory on write transactions. 0 Enable memory error correction. 1 Disable memory error correction.
24 NCEFAFRZ	Non-Correctable Errors In FlexCAN Access Put Device In Freeze Mode Determines the response when a non-correctable error is detected in a memory read performed by FlexCAN internal processes. In this case, entering Freeze mode prevents corrupted data from being treated as valid by FlexCAN internal processes. 0 Keep normal operation. 1 Put FlexCAN in Freeze mode (according to the "Freeze mode" section).
25–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.3.21 Error Injection Address Register (CAN_ERRIAR)

This register holds the address where error is to be injected.

Use the following table to convert from the memory map address to the location in the physical FlexCAN RAM:

RAM contents	Injection address	Memory map
FlexCAN Registers	Not mapped	-
MBs	0x0000	0x0080
Reserved	-	0x0480
RXIMRs	0x0400	0x0880
Reserved	-	0x0980
RXFIR_0	0x0500	0x0A80
RXFIR_1	0x0504	0x0A84
RXFIR_2	0x0508	0x0A88
RXFIR_3	0x050C	0x0A8C
RXFIR_4	0x0510	0x0A90
RXFIR_5	0x0514	0x0A94
Reserved	-	0x0A98

Table continues on the next page...

RAM contents	Injection address	Memory map
RXMGMASK	0x0520	0x0AA0
RXFGMASK	0x0524	0x0AA4
RX14MASK	0x0528	0x0AA8
RX15MASK	0x052C	0x0AAC
SMB_TX	0x0530	0x0AB0
SMB_RX0	0x0540	0x0AC0
SMB_RX1	0x0550	0x0AD0
ECC Registers	Not mapped	0x0AE0
Reserved	-	0x0B00

Address: 0h base + AE4h offset = AE4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																		INJADDR													
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIAR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 INJADDR	Address Where Error Is To Be Injected The two least significant bits of this field always read as 0.

45.3.22 Error Injection Data Pattern Register (CAN_ERRIDPR)

Holds the error pattern to be injected in the data word read from memory.

Address: 0h base + AE8h offset = AE8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DFLIP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIDPR field descriptions

Field	Description
0–31 DFLIP	Data flip pattern Bits set to 1 in the flip pattern cause the corresponding data bit in the word read from memory to invert.

45.3.23 Error Injection Parity Pattern Register (CAN_ERRIPPR)

Holds the error pattern to be injected in parity bits read from memory along with data word.

Bits set to 1 in the flip pattern cause the corresponding parity bit in the word read from memory to invert.

Address: 0h base + AECh offset = AECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			PFLIP3					0			PFLIP2					0			PFLIP1					0			PFLIP0				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIPPR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PFLIP3	Parity Flip Pattern For Byte 3 (most significant)
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PFLIP2	Parity Flip Pattern For Byte 2
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 PFLIP1	Parity Flip Pattern For Byte 1
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PFLIP0	Parity Flip Pattern For Byte 0 (Least Significant)

45.3.24 Error Report Address Register (CAN_RERRAR)

Reports the address used for an access in which an error (correctable or non-correctable) was detected, and also reports the identification of the source of that access.

This address is always reported using a 32-bit alignment. Non-aligned accesses (ERRADDR[1:0] non-0) are reported with the address aligned and data is reported in RERRDR accordingly shifted. In case of errors detected in accesses larger than 32-bit (as performed by FlexCAN internal processes), the address of the 32-bit word in which the error was detected is reported. In case of errors detected in more than one 32-bit word, only the least significant address is reported.

Address: 0h base + AF0h offset = AF0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							NCE	0					SAID		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ERRADDR													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RERRAR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–6
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>7
NCE</td><td>Non-Correctable Error
Indicates that the report is due to an non-correctable error.
0 Reporting a correctable-error
1 Reporting a non-correctable error</td></tr>
<tr><td>8–12
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>13–15
SAID</td><td>SAID[2] — Identification of the requestor of the memory read request:
• 0 = Requested by FlexCAN internal processes
• 1 = Requested by Host (CPU)
SAID[1] — Details of FlexCAN operation:
• 0 = Move
• 1 = Scanning
SAID[0] — Operation that requested the memory read:
• 0 = Transmission
• 1 = Reception
<b>Table 45-3. Source of memory access</b>
<table>
<tr><th>SAID[2:0]</th><th>Error during...</th></tr>
<tr><td>0</td><td>Tx Move</td></tr>
<tr><td>1</td><td>Rx Move</td></tr>
<tr><td>2</td><td>Tx Arbitration</td></tr>
<tr><td>3</td><td>Rx Match</td></tr>
<tr><td>4</td><td>Tx Move</td></tr>
<tr><td>5-7</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>16–17
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
</table>

Table continues on the next page...

CAN_RERRAR field descriptions (continued)

Field	Description
18–31 ERRADDR	Address Where The Error Was Detected See the description of the Error Injection Address Register (ERRIAR).

45.3.25 Error Report Data Register (CAN_RERRDR)

Reports the raw data (unmodified by the correction performed by ECC logic) read from memory with error. The value reported does not represent the transient values of the BUSY bit (see the “Message Buffer Code for Rx buffers” table) when reading a Message Buffer.

Address: 0h base + AF4h offset = AF4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RERRDR field descriptions

Field	Description
0–31 RDATA	Raw data word read from memory with error

45.3.26 Error Report Syndrome Register (CAN_RERRSYNR)

Holds the syndrome detected in a memory read with error. It also reports the bytes which were read in this 32-bit read transaction.

Each SYNDn field indicates the type of error and which bit in byte (n) is affected by the error. (SYND3 corresponds to the most significant byte in the data word read from memory; SYND0 corresponds to the least significant.)

Table 45-4. Syndrome Definition

SYNDn (hex)	Type	Bit affected
00	-	none (no error)
01	Code	0
02	Code	1

Table continues on the next page...

Table 45-4. Syndrome Definition (continued)

SYNDn (hex)	Type	Bit affected
04	Code	2
07	Data	5
08	Code	3
0E	Data	7
10	Code	4
13	Data	2
15	Data	6
16	Data	1
19	Data	3
1A	Data	4
1C	Data	0
All others	-	Non-correctable error

Each BEn field indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated (see the "Error Indication" section) and reported (see the "Error Reporting" section).

Address: 0h base + AF8h offset = AF8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BE3	0		SYND3					BE2	0		SYND2				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE1	0		SYND1					BE0	0		SYND0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RERRSYNR field descriptions

Field	Description
0 BE3	Byte Enabled For Byte 3 (Most Significant) 0 The byte was not read. 1 The byte was read.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 SYND3	Error Syndrome For Byte 3 (Most Significant) See the "Syndrome Definition" table.

Table continues on the next page...

CAN_RERRSYNR field descriptions (continued)

Field	Description
8 BE2	Byte Enabled For Byte 2 0 The byte was not read. 1 The byte was read.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SYND2	Error Syndrome For Byte 2 See the "Syndrome Definition" table.
16 BE1	Byte Enabled For Byte 1 0 The byte was not read. 1 The byte was read.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYND1	Error Syndrome for Byte 1 See the "Syndrome Definition" table.
24 BE0	Byte Enabled For Byte 0 (least significant) 0 The byte was not read. 1 The byte was read.
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 SYND0	Error Syndrome For Byte 0 (least significant) See the "Syndrome Definition" table.

45.3.27 Error Status Register (CAN_ERRSR)

Holds the status bits of the error correction and detection operations. After raised, these flags must be cleared by writing 1 to them. Writing 0 has no effect.

Address: 0h base + AFCh offset = AFCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												HANCEIF	FANCEIF	0	CEIF
W													w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												HANCEIOF	FANCEIOF	0	CEIOF
W													w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 HANCEIF	Host Access With Non-Correctable Error Interrupt Flag Indicates that a non-correctable error was detected in a memory access initiated by Host. A bus transfer error is asserted for that access. No interrupt is associated to this flag. 0 No non-correctable errors were detected in Host accesses so far. 1 A non-correctable error was detected in a Host access.
13 FANCEIF	FlexCAN Access With Non-Correctable Error Interrupt Flag

Table continues on the next page...

CAN_ERRSR field descriptions (continued)

Field	Description
	Indicates that a non-correctable error was detected in a memory access initiated by FlexCAN internal processes. No interrupt is associated to this flag. 0 No non-correctable errors were detected in FlexCAN accesses so far. 1 A non-correctable error was detected in a FlexCAN access.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CEIF	Correctable Error Interrupt Flag Indicates that a correctable error was detected in a memory access. If MECR[CEI_MSK] is set, the interrupt is asserted. 0 No correctable errors were detected so far. 1 A correctable error was detected.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 HANCEIOF	Host Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory access initiated by Host when HANCEIF was set. No interrupt is associated to this flag. (See the "Error Indication" section.) 0 No overrun on non-correctable errors in Host access 1 Overrun on non-correctable errors in Host access
29 FANCEIOF	FlexCAN Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory access initiated by FlexCAN internal processes when FANCEIF was set. No interrupt is associated to this flag. (See the "Error Indication" section.) 0 No overrun on non-correctable errors in FlexCAN access 1 Overrun on non-correctable errors in FlexCAN access
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEIOF	Correctable Error Interrupt Overrun Flag Indicates that a correctable error was detected in a memory access when CEIF was set. No interrupt is associated to this flag. (See the "Error Indication" section.) 0 No overrun on correctable errors 1 Overrun on correctable errors

45.3.28 Message buffer structure

The message buffer structure used by the FlexCAN module is represented in the following figure. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16 bytes.

The memory area from 0x80 to 0x47F is used by the mailboxes.

Table 45-5. Message buffer structure

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>23</th><th>24</th><th>31</th></tr>
<tr><td>0x0</td><td colspan="4">▨</td><td colspan="2">CODE</td><td>▨</td><td>SRR</td><td>IDE</td><td>RTR</td><td colspan="4">DLC</td><td colspan="4">TIME STAMP</td></tr>
<tr><td>0x4</td><td colspan="3">PRIO</td><td colspan="9">ID (Standard/Extended)</td><td colspan="6">ID (Extended)</td></tr>
<tr><td>0x8</td><td colspan="6">Data Byte 0</td><td colspan="8">Data Byte 1</td><td colspan="2">Data Byte 2</td><td colspan="2">Data Byte 3</td></tr>
<tr><td>0xC</td><td colspan="6">Data Byte 4</td><td colspan="8">Data Byte 5</td><td colspan="2">Data Byte 6</td><td colspan="2">Data Byte 7</td></tr>
</table>

▨ = Unimplemented or Reserved

CODE — Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 45-6 and Table 45-7. See Functional description for additional information.

Table 45-6. Message buffer code for Rx buffers

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0000: INACTIVE — MB is not active.	INACTIVE	—	—	—	MB does not participate in the matching process.
0b0100: EMPTY — MB is active and empty.	EMPTY	—	FULL	—	When a frame is received successfully (after the Move-in) process), the CODE field is automatically updated to FULL.
0b0010: FULL — MB is full.	FULL	Yes	FULL	—	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching process for matching details related to FULL code.

Table continues on the next page...

Table 45-6. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
		No	OVERRUN	—	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching process for details about overrun behavior.
0b0110: OVERRUN — MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	—	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to FULL.
		No	OVERRUN	—	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching process for details about overrun behavior.
0b1010: RANSWER[4] — A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.	RANSWER	—	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that an MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). See Matching process for details. If CTRL2[RRS] is negated, transmit a response frame whenever a remote

Table continues on the next page...

Table 45-6. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
					request frame with the same ID is received.
		—	—	1	This code is ignored during matching and arbitration process. See Matching process for details.
CODE[0]=1b1: BUSY — FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[5]	—	FULL	—	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		—	OVERRUN	—	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered a successful reception after the frame to be moved to MB (move-in process). See Move-in for details.
3. Remote Request Stored bit from CTRL2 register. See section "Control 2 Register (CTRL2)" for details.
4. Code 0b1010 is not considered Tx and an MB with this code should not be aborted.
5. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 45-7. Message buffer code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE — MB is not active	INACTIVE	—	—	MB does not participate in arbitration process.
0b1001: ABORT — MB is aborted	ABORT	—	—	MB does not participate in arbitration process.
0b1100: DATA — MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE — MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER — MB is a Tx Response	TANSWER	—	RANSWER	This is an intermediate code that is

Table 45-7. Message buffer code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
Frame from an incoming Remote Request Frame				automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching process and Arbitration process for details.

SRR — Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extended format frames

IDE — ID Extended Bit

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

RTR — Remote Transmission Request

This bit affects the behavior of remote frames and is part of the reception filter. See Table 45-6, Table 45-7, and the description of the RRS bit in Control 2 Register (CTRL2) for additional details.

If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted. In Rx MB it may be considered in matching processes.

DLC — Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see the Table 45-5). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 45-8).

TIME STAMP — Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO — Local priority

This 3-bit field is used only when LPRIO_EN bit is set in MCR, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID — Frame Identifier

In standard frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0–7 — Data Field

Up to eight bytes can be used for a data frame.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC as shown in the table below.

For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 45-8. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0–1
3	DATA BYTE 0–2
4	DATA BYTE 0–3
5	DATA BYTE 0–4
6	DATA BYTE 0–5
7	DATA BYTE 0–6
8	DATA BYTE 0–7

45.3.29 Rx FIFO structure

When the MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6–37) depending on the CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Rx FIFO data structure.

Table 45-9. Rx FIFO structure

<table>
<tr><th></th><th>0</th><th>3</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>23</th><th>24</th><th>31</th></tr>
<tr><td>0x80</td><td colspan="4"></td><td>SRR</td><td>IDE</td><td>RTR</td><td colspan="4">DLC</td><td colspan="4">TIME STAMP</td></tr>
<tr><td>0x84</td><td></td><td colspan="8">ID standard</td><td colspan="6">ID extended</td></tr>
<tr><td>0x88</td><td colspan="3">Data byte 0</td><td colspan="8">Data byte 1</td><td colspan="2">Data byte 2</td><td colspan="2">Data byte 3</td></tr>
<tr><td>0x8C</td><td colspan="3">Data byte 4</td><td colspan="8">Data byte 5</td><td colspan="2">Data byte 6</td><td colspan="2">Data byte 7</td></tr>
<tr><td>0x90
to
0xDC</td><td colspan="15">Reserved</td></tr>
</table>

Table continues on the next page...

Table 45-9. Rx FIFO structure (continued)

0xE0	ID filter table element 0
0xE4	ID filter table element 1
0xE8 to 0x2D4	ID filter table elements 2 to 125
0x2D8	ID filter table element 126
0x2DC	ID filter table element 127
▨	= Unimplemented or reserved

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Table 45-10. ID table structure

Format	0	1	2 … 7	8 … 15	16	17	18 … 23	24 … 30	31
A	RTR	IDE	RXIDA (standard = 2–12, extended = 2–30)						▨
B	RTR	IDE	RXIDB_0 (standard = 2–12, extended = 2–15)		RTR	IDE	RXIDB_1 (standard = 18–28, extended = 18–31)		
C	RXIDC_0 (std/ext = 0–7)			RXIDC_1 (std/ext = 8–15)	RXIDC_2 (std/ext = 16–23)			RXIDC_3 (std/ext = 24–31)	
	▨		= Unimplemented or Reserved						

RTR — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

IDE — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (2 to 12) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (2 to 12 and 18 to 28) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

45.4 Functional description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message buffer structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 45-6). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 45-7).

45.4.1 Transmit process

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission abort mechanism). If backwards compatibility is desired (MCR[AEN] bit is negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Mailbox inactivation).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control, and CODE fields of the Control and Status word to activate the MB.

When the MB is activated, it will participate into the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the CODE field in the Control and Status word is updated, the CRC Register is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB (see Table 45-6 and Table 45-7 in Message buffer structure).

When the Abort feature is enabled (MCR[AEN] is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked, therefore the CPU is not able to update it until the Interrupt Flag is negated by CPU. This means that the CPU must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

45.4.2 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest number Mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CTRL2[TASD] field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CTRL1[LBUF] and MCR[LPRIOEN] bits settings.

45.4.2.1 Lowest-number Mailbox first

If CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. MCR[LPRIOEN] bit has no effect when CTRL1[LBUF] is asserted.

45.4.2.2 Highest-priority Mailbox first

If CTRL1[LBUF] bit is negated, then the arbitration process searches the active Tx Mailbox with the highest priority, which means that this Mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest-priority Tx Mailbox is the one that has the lowest arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values, the Mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIOEN] bit setting.

45.4.2.2.1 Local Priority disabled

If MCR[LPRIOEN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the Local Priority is disabled.

Table 45-11. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

45.4.2.2.2 Local Priority enabled

If Local Priority is desired MCR[LPRIOEN] must be asserted. In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the following table).

Table 45-12. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

45.4.2.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in Freeze mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the Data Length Code (DLC) field value is greater than that.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. TASD value may be changed to optimize the arbitration start point.
- During CAN BusOff state from TX_ERR_CNT=124 to 128. TASD value may be changed to optimize the arbitration start point.
- During C/S write by CPU in BusIdle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after arbitration process has finished, then TX arbitration machine begins a new arbitration process.
    - If there is a pending arbitration and BusIdle state starts then an arbitration process is triggered. In this case the first and second C/S write in BusIdle will not restart the arbitration process. It is possible that there is not enough time to finish arbitration in WaitForBusIdle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during BusIdle state. During this arbitration C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon Leave Freeze mode (first bit of the WaitForBusIdle state). If there is a re-synchronization during WaitForBusIdle arbitration process is restarted.

Arbitration process stops in the following situation:

- All Mailboxes were scanned
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled
- Arbitration winner inactivation or abort during any arbitration process
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus
- Low Power or Freeze mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer
- Any C/S write if there is no Tx Arbitration process in progress
- Rx Match has just updated a Rx Code to Tx Code
- Entering Busoff state

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in a MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

45.4.3 Receive process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Mailbox inactivation), preferably with a safe inactivation (see Transmission abort mechanism).

2. Write the ID word

3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in* process (see Move-in) as follows:

1. The received Data field (8 bytes at most) is stored.
2. The received Identifier field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox's Time Stamp field.
4. The received SRR, IDE, RTR, and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 45-6 and Table 45-7 in Section Message buffer structure).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for CPU servicing (read) the frame received in an Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox.
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Mailbox lock mechanism.
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in.
4. Acknowledge the proper flag at IFLAG registers.
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should synchronize to frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 45-6 . If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received frame matching the filter of that Mailbox may be lost.

CAUTION

In summary: *never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.*

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in an Mailbox may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists a matching Rx Mailbox, provided the MCR[SRXDIS] bit is not asserted. If the MCR[SRXDIS] bit is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of the IFLAG[BUF5I] "Frames available in Rx FIFO" bit in the IMASK1 register), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional – needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional – needed only if a mask was used)
3. Read the Data field
4. Read the RXFIR register (optional)
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to IFLAG[BUF5I] bit (mandatory – releases the MB and allows the CPU to read the next Rx FIFO entry)

45.4.4 Matching process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. In any case, the matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame

- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the SMB will be transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results will be transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. See the following table.

Table 45-13. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Mailbox	0	-	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

[1] For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on IDAM.
[2] cmp: Compares the SMB contents with the MB contents regardless the masks.
[3] no_cmp: The SMB contents are not compared with the MB contents
[4] cmp_msk: Compares the SMB contents with MB contents taking into account the masks.
[5] SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- The CODE field of the Mailbox is EMPTY

- The CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in Mailbox lock mechanism)
- The CODE field of the Mailbox is either FULL or OVERRUN and an inactivation (see Mailbox inactivation) is performed
- The Rx FIFO is not full

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the MCR[IRMQ] bit. If it is negated the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive then the Rx FIFO is the matching winner regardless of the scan for Mailboxes
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above

If the selected priority is Mailboxes first:

- If a free-to-receive matched Mailbox is found, it is the matching winner regardless the scan for Rx FIFO
- If no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO
- If both conditions above are not satisfied and a non free-to-receive matched Mailbox is found then the matching winner determination is conditioned by the MCR[IRMQ] bit:
    - If MCR[IRMQ] bit is negated the matching winner is the first matched Mailbox
    - If MCR[IRMQ] bit is asserted the matching winner is the Rx FIFO if it is a free-to-receive matched structure, otherwise the matching winner is the last non free-to-receive matched Mailbox

See the following table for a summary of matching possibilities.

Table 45-14. Matching possibilities and resulting reception structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	-[3]	None	Frame lost by no match
0	0	X	Free[4]	-	FirstMB	
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMb	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMb	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMb	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMb	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMb	Overrun

[1] This is a don't care condition.

[2] Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).

3. This is a forbidden condition.
4. Matched in MB “Free” means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO “None” means that the frame has not matched any filter in FIFO. It is as if the FIFO didn’t exist (CTRL2[RFEN]=0).
6. Matched in FIFO “NotFull” means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO “Full” means that the frame has matched a FIFO filter but couldn’t store it because it has no empty slots to receive it.

If a non-safe Mailbox inactivation (see Mailbox inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not "free-to-receive", so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Refer to the description of the Rx Individual Mask Registers (RXIMRx). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care". Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze mode; otherwise, they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (RXGMASK, RX14MASK, RX15MASK and RXFGMASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the MCR Register is negated.

45.4.5 Move process

There are two types of move process: move-in and move-out.

45.4.5.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching process) and all of the following conditions are true:

- The CAN bus has reached or let past either:
  - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB
- There is no ongoing matching process
- The destination Mailbox is not locked by the CPU
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished
- There is a previous pending move-in to the same destination Mailbox
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled (MCR[SRXDIS] bit is asserted)
- Any CAN protocol error is detected

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It only stays on hold waiting for exiting Freeze and Low-Power mode and to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. if the message is destined to the Rx FIFO, push IDHIT into the RXFIR FIFO;
2. reads the words DATA0-3 and DATA4-7 from the Rx SMB;
3. writes it in the words DATA0-3 and DATA4-7 of the Rx Mailbox;
4. reads the words Control/Status and ID from the Rx SMB;
5. writes it in the words Control/Status and ID of the Rx Mailbox, updating the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Mailbox inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed to alert the CPU that the Message Buffer content is temporarily incoherent.

45.4.5.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Section "Arbitration process"). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

45.4.6 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Transmit process and Receive process.

45.4.6.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- MCR[AEN] bit must be asserted
- The first CPU action must be the writing of abort code (0b1001) into the CODE field of the Control and Status word.

The active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to a Mailbox that is currently being transmitted, or to a Mailbox that was already loaded into the SMB for transmission, the write operation is blocked and the MB is kept active, but the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze mode
- The module enters in BusOff state
- There is an overload frame

If none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the interrupt flag is set in the IFLAG, and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was *safely inactivated*. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- CPU checks the corresponding IFLAG and clears it, if asserted.

- CPU writes 0b1001 into the CODE field of the C/S word.
- CPU waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- CPU reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
- It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.

45.4.6.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on Mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission Mailboxes must be performed just when MCR[AEN] bit is deasserted.

If a Mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See Transmit process and Receive process for more detailed instruction on how to inactivate and reactivate a Mailbox.

To inactivate a Mailbox, the CPU must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

Because the user is not able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can lead to undesirable results:

- A frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter
- A frame containing the message within the inactivated Tx Mailbox may be transmitted without notice

In order to eliminate such risk and perform a *safe inactivation* the CPU must use the following mechanism along with the inactivation itself:

- For Tx Mailboxes, the Transmission Abort (see Transmission abort mechanism)

The inactivation automatically unlocks the Mailbox (see Mailbox lock mechanism).

NOTE

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by

FlexCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

45.4.6.3 Mailbox lock mechanism

Other than Mailbox inactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code. A CPU write into C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending-move and potentially may lose a received message. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

NOTE

The locking mechanism applies only to Rx MBs that are not part of FIFO and have a code different than INACTIVE (0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

[1] In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

Note

> If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see Section "Free Running Timer Register (TIMER)"), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see Modes of operation) and it will take place only when the module resumes to Normal or Freeze modes.

45.4.7 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the MCR[RFEN] bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature. The FIFO is 6-message deep. The memory region occupied by the FIFO structure (both Message Buffers and FIFO engine) is described in Rx FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The IFLAG[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a Message Buffer) and the RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is only valid whilst the IFLAG[BUF5I] is asserted.

The IFLAG[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

The IFLAG[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Note

> A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read by accessing the RXFIR register. The RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid while the IFLAG[BUF5I] flag is asserted. The RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (RXIMRx), according to the setting of CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the IRMQ bit is negated, then the FIFO filter table is affected by RXFGMASK.

45.4.8 CAN protocol related features

This section describes the CAN protocol related features.

45.4.8.1 Remote frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by writing the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame was received and matched a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

45.4.8.2 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

45.4.8.3 Time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

The Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in the description of the Control 1 Register (CTRL1).

45.4.8.4 Protocol timing

The following figure shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule. The clock source bit CLKSRC in the CTRL1 Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock. In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (bit MDIS set in the Module Configuration Register).

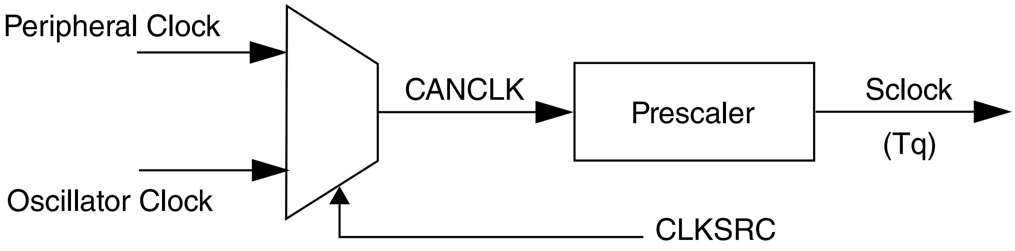


Figure 45-2. CAN engine clocking scheme

The oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The oscillator clock has better jitter performance.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See the description of the Control 1 Register (CTRL1).

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler Value})}$$

A bit time is subdivided into three segments[1] (see Figure 45-3 and Table 45-15):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL1 Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL1 Register (plus 1) to be 2 to 8 time quanta long

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

[1] For further explanation of the underlying concepts, see ISO/DIS 11519–1, Section 10.3. See also the CAN 2.0A/B protocol specification for bit timing.

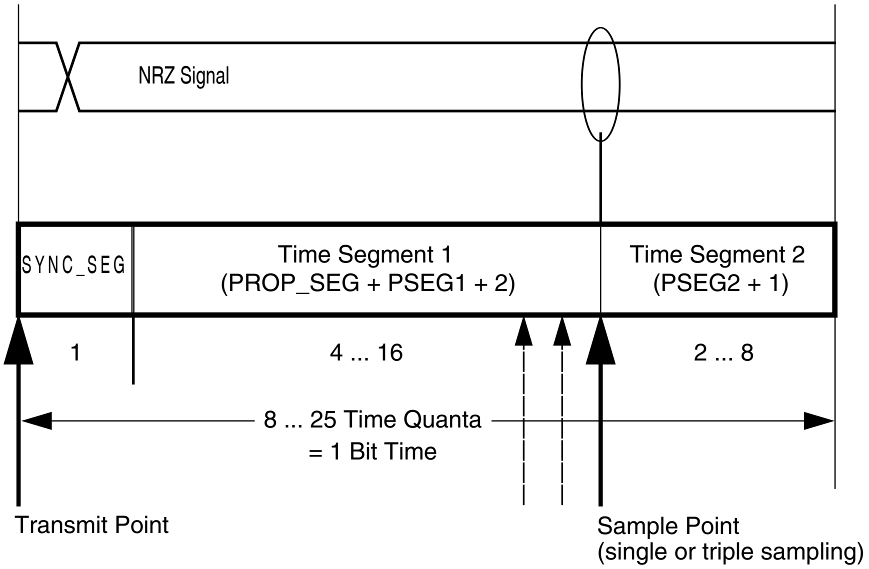


Figure 45-3. Segments within the bit time

Whenever CAN bit is used as a measure of duration (for example, MCR[FRZACK] and MCR[LPMACK]), the number of peripheral clocks in one CAN bit can be calculated as:

$$\mathrm{NCCP} = \frac{f_{SYS} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)}{f_{CANCLK}}$$

where:

- NCCP is the number of peripheral clocks in one CAN bit;
- f_{CANCLK} is the Protocol Engine (PE) Clock (see Figure "CAN Engine Clocking Scheme"), in Hz;
- f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;
- PSEG1 is the value in CTRL1[PSEG1] field;
- PSEG2 is the value in CTRL1[PSEG2] field;
- PROPSEG is the value in CTRL1[PROPSEG] field;
- PRESDIV is the value in CTRL1[PRESDIV] field.

For example, 180 CAN bits = 180 x NCCP peripheral clock periods.

Table 45-15. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The following table gives an overview of the CAN compliant segment settings and the related parameter values.

Table 45-16. Bosch CAN 2.0B standard compliant bit time segment settings

Time segment 1	Time segment 2	Re-synchronization jump width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

Note

The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1). For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

45.4.8.5 Arbitration and matching timing

During normal reception and transmission of frames, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

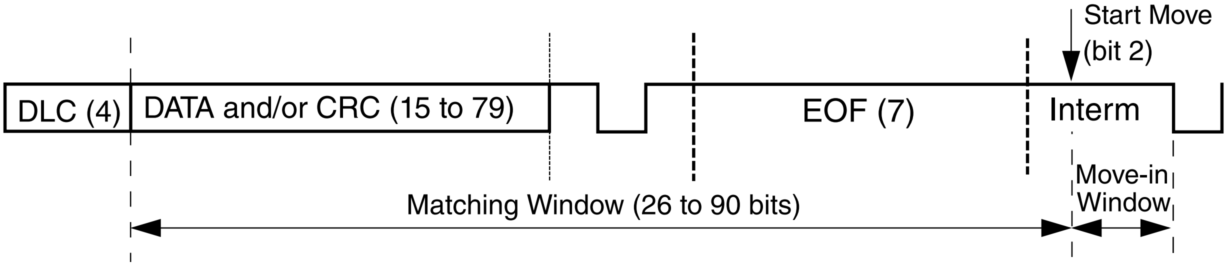


Figure 45-4. Matching and move-in time windows

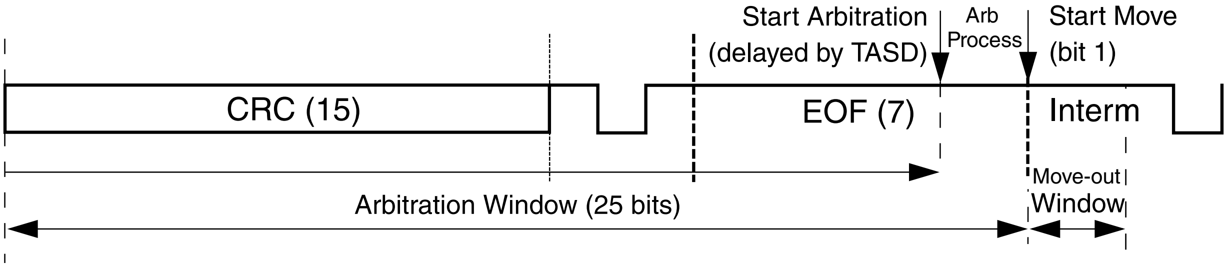


Figure 45-5. Arbitration and move-out time windows

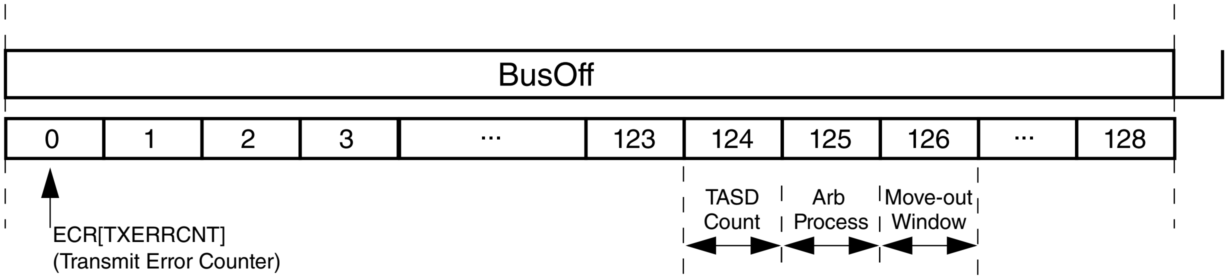


Figure 45-6. Arbitration at the end of bus off and move-out time windows

NOTE

The matching and arbitration timing shown in the preceding figures do not take into account the delay caused by the concurrent memory access due to the CPU or other internal FlexCAN sub-blocks.

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 45-16
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, see Clock domains and restrictions
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in the following table:

Table 45-17. Minimum ratio between peripheral clock frequency and CAN bit rate

Number of Message Buffers	RFEN	Minimum number of peripheral clocks per CAN bit
16 and 32	0	16
64	0	25
16	1	16
32	1	17
64	1	30

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, therefore the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in the preceding table can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2) contained in the Control 1 Register (CTRL1).

In case of synchronous operation (when the peripheral clock frequency is equal to the oscillator clock frequency), the number of peripheral clocks per CAN bit can be adjusted by selecting an adequate value for PRESDIV in order to meet the requirement in the preceding table. In case of asynchronous operation (the peripheral clock frequency greater than the oscillator clock frequency), the number of peripheral clocks per CAN bit can be adjusted by both PRESDIV and/or the frequency ratio.

As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

45.4.9 Clock domains and restrictions

The FlexCAN module has two clock domains asynchronous to each other:

- The Bus Domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock.
- The Oscillator Domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the CAN bus.

When CTRL1[CLKSRC] bit is set, synchronous operation occurs because both domains are connected to the peripheral clock (creating a 1:1 ratio between the peripheral and oscillator domain clocks).

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the Bus Domain clock frequency must always be greater than the Oscillator Domain clock frequency.

NOTE

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

45.4.10 Modes of operation details

The FlexCAN module has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following sub-sections contain functional details on Freeze mode and the low-power modes.

CAUTION

> "Permanent Dominant" failure on CAN Bus line is not supported by FlexCAN. If a Low-Power request or Freeze mode request is done during a "Permanent Dominant", the corresponding acknowledge can never be asserted.

45.4.10.1 Freeze mode

This mode is requested by the CPU through the assertion of the HALT bit in the MCR Register or when the chip is put into Debug mode. In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in a low-power mode. The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The CPU must only consider the FlexCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CTRL1[CLK_SRC] bit that can be read but cannot be written.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the MCR Register
- The chip is removed from Debug Mode and/or the HALT bit is negated

The FRZ_ACK bit is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

45.4.10.2 Module Disable mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the CPU through the assertion of the MDIS bit in the MCR Register and the acknowledgement is obtained through the assertion by the FlexCAN of the LPM_ACK bit in the same register. The CPU must only consider the FlexCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit.

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and negate the LPM_ACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

45.4.10.3 Stop mode

This is a system low-power mode in which all chip clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and the acknowledgement is obtained through the assertion by the FlexCAN of a Stop Acknowledgement signal. The CPU must only consider the FlexCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it sets the LPMACK bit, negates the FRZACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOTRDY and LPMACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop Mode request.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FlexCAN negates the LPMACK bit. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up.

45.4.11 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Error, ECC Host Access Non-correctable Error, ECC FlexCAN Access Non-correctable Error, ECC Correctable Error, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception.

Each of the buffers has assigned a flag bit in the IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes it to 1 (unless another interrupt is generated at the same time).

Note

> It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (MCR[RFEN] = 1), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of IFLAG1 becomes the "FIFO Overflow" flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the "Frames Available in FIFO flag," and bits 4-0 are unused. See the description of IFLAG1 for more information.

For a combined interrupt where multiple MB interrupt sources are ORed together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Error, Tx Warning, and Rx Warning generate interrupts like the MB interrupt sources, and they can be read from ESR1 and ESR2. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in CTRL1.

The interrupt sources for ECC errors—Host Access with Non-correctable Interrupt Flag, FlexCAN Access with Non-correctable Interrupt Flag, and Correctable Error Interrupt Flag—can be read in ERRSR and the corresponding interrupts masks bits are located in MECR.

45.4.12 Bus interface

The CPU access to FlexCAN registers are subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in Table "Module Memory Map" in Supervisor Mode or with S only) results in access error.
- Read and write access to implemented reserved address space results in access error.

- Write access to positions whose bits are all currently read-only results in access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to register and bit descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power Mode results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 16 MBs, RFFN is 0x0, and MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one Individual Mask and the available memory in the Mask Registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

45.4.13 Detection and Correction of Memory Errors

FlexCAN supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits and the error correction mechanism assures that in this 13-bit word, errors in one bit can be corrected (corrected errors) and errors in 2 bits can be detected but not corrected (non-corrected errors). Errors in more than 2 bits may not be detected. When read, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

Memory errors are indicated to Host through status register (Error Status Register (ERRSR)) and bus transfer errors, and reported through report registers (Error Report Address Register (RERRAR), Error Report Data Register (RERRDR) and Error Report Syndrome Register (RERRSYNR)).

The error detection and correction mechanism can be activated or not, controlled by ECCDIS bit in Memory Error Control Register (MECR). When disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations. It assures that memory has consistent parity bits associated to data.

NOTE

All FlexCAN memory must be initialized before starting its operation in order to have the parity bits in memory properly updated. The WRMFRZ bit in Control 2 Register (CTRL2) grants write access to all memory positions from 0x080 to 0xADF.

To avoid that critical error correction configuration be accidentally changed, this protocol must be followed to enable the update of the Memory Error Control Register (MECR):

1. By default, ECRWRE bit in Control 2 Register (CTRL2) is 0 and ECRWRDIS bit in Memory Error Control Register (MECR) is 1.
2. Set ECRWRE bit in Control 2 Register (CTRL2).
3. Clear ECRWRDIS bit.
4. All writes to Memory Error Control Register (MECR) must keep ECRWRDIS cleared.
5. After configuration is done, lock the Memory Error Control Register (MECR) by either setting ECRWRDIS or clearing ECRWRE.

45.4.13.1 Sources of the Memory Access

The FlexCAN memory can be accessed by 2 major sources (or requestors):

- by Host (CPU): largest word accessed is 32-bit wide
- by FlexCAN internals processes (Match, Arbitration, RxMove on reception, TxMove on transmission): largest word accessed is 64-bit wide

The way that non-correctable errors are indicated and reported depends on the source of access.

45.4.13.2 Error Indication

Memory errors are indicated by flags HANCEIF, FANCEIF, CEIF in the Error Status Register (ERRSR). Non-correctable errors detected in memory reads requested by Host are indicated separatedly than the ones detected in requests by FlexCAN internal processes. Corrected errors indication makes no distinction of the source of the access. There are 3 independent flags for these 3 cases, and each flag will raise an interrupt unless it is masked by mask bits in Memory Error Control Register (MECR). If both non-corrected and corrected errors are found in different bytes in the same read operation both flags are set.

The non-corrected error detected in Host accesses are also indicated as a bus transfer error. A bus wait request may be asserted to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked. If the flag bit HANCEIF is not masked, the same non-corrected error will raise a bus transfer error and an interrupt request.

Each indication flag has one Overrun flag in Error Status Register (ERRSR). The Overrun flags do not request interrupts. Overrun flags for non-correctable errors indicate that other errors of the same nature were detected after current error being treated, while overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated. This is the recommended handling sequence for error indication:

1. Get error report information from report registers
2. Use this information to take proper measures in the application
3. Clear the Interrupt flag
4. If the Overrun flag is active:
    a. Alert application that at least one error could not be handled
    b. Clear the Overrun flag

The FlexCAN internal processes can access memory in transactions larger than 32-bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in 2 or more 32-bit words the Interrupt and Overrun flags are set simultaneously.

45.4.13.3 Error Reporting

Report registers Error Report Address Register (RERRAR), Error Report Data Register (RERRDR) and Error Report Syndrome Register (RERRSYNR) provide detailed information about the address read, raw data and syndrome read with error and indicated by the flags described in the “Error Indication” section. The address, data and syndrome registers are updated simultaneously along with the error flags, according to these rules.

1. If any of the 2 non-corrected error flags is currently set, does not update the registers (preserve the old non-corrected error reporting)
2. Else (no error flag is currently set or only the corrected error flag is currently set), update the report registers according to the new error; or according to the most severe of new errors if non-corrected and corrected errors are simultaneously detected

Reporting of errors detected in accesses larger than 32-bit follows the semantic described in the “Error Indication” section and in the Error Report Address Register (RERRAR) description.

The address reported in RERRAR and defined in ERRIAR are not the same listed in the module memory map. How the reported addresses correspond to locations in the module memory map is shown in the Error Injection Address Register (ERRIAR) description.

Addresses reported when reading memory portions organized as FIFOs, such as the Rx FIFO Structure and the Rx FIFO Information Register (RXFIR), refer to the address of the element accessed in the FIFO, not to the FIFO base address.

To assure coherence of the error report registers it is necessary to turn off the report update by clearing the MECR[RERRDIS] bit before reading the report registers.

45.4.13.4 Response to Errors

Correctable errors have no consequence on FlexCAN operation, once the corrected data is used by Host or FlexCAN internal processes. For host-initiated reads, a non-corrected error detected does not affect FlexCAN operation.

Non-corrected errors detected on memory reads requested by FlexCAN do not stop the FlexCAN processing and this may result in incorrect operation.

If a non-corrected error occurs during a reception process (Match or RxMove), an incorrect destination may be selected to store the incoming frame, a corrupted frame may be stored in the correct destination, or both. If a non-corrected error occurs during either an Arbitration or a TxMove process, either a non-highest priority frame may be mistakenly selected to be transmitted or no frame may be transmitted at all, when MECR[NCEFAFRZ] = 1.

The error report registers can provide information to the application for a customized handling of these situations. If the delay of this handling is not accepted and MECR[NCEFAFRZ] = 1, the FlexCAN stops operation automatically and enters Freeze mode when these errors are detected. See the Memory Error Control Register (MECR) description for more details.

45.4.13.5 Error Injection

The error injection registers ERRIAR, ERRIDPR, and ERRIPPR are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers. How the error injection addresses correspond to locations in the module memory map is shown in the ERRIAR description.

The injection is done by flipping the data and parity bits correspondent to the bits in 1 in ERRIDPR and ERRIPPR. Injection can be selected specifically for memory accesses requested by Host or by FlexCAN internal processes.

In case of accesses larger than 32-bits, the EXTERRIE bit in Memory Error Control Register (MECR) extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

NOTE

> It is very unlikely, but error injection may correct a bit with error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, the HAERRIE and FAERRIE bits in Memory Error Control Register (MECR) must be cleared while configuring the memory injection registers.

45.5 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

45.5.1 FlexCAN initialization sequence

The FlexCAN module may be reset as follows:

- Chip level hard reset, which resets all memory mapped registers asynchronously
- SOFTRST bit in MCR, which resets some of the memory mapped registers synchronously. See Table 45-2 to see what registers are affected by soft reset.
- Chip level soft reset, which has the same effect as the SOFTRST bit in MCR

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFTRST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in a low power mode. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLKSRC bit) should be selected while the module is in Disable mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze mode. In Freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze mode; see Freeze mode. The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register
    - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
    - Enable the warning interrupts by setting the WRNEN bit
    - If required, disable frame self reception by setting the SRXDIS bit
    - Enable the Rx FIFO by setting the RFEN bit
    - Enable the abort mechanism by setting the AEN bit
    - Enable the local priority feature by setting the LPRIOEN bit
- Initialize the Control Register
    - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
    - Determine the bit rate by programming the PRESDIV field
    - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
    - The Control and Status word of all Message Buffers must be initialized
    - If Rx FIFO was enabled, the ID filter table must be initialized
    - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- When ECC is enabled, follow the initialization note in Detection and Correction of Memory Errors
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts) and in CTRL Register (for Bus Off and Error interrupts)
- Negate the HALT bit in MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

Chapter 46
Zipwire

46.1 Overview

The SIPI and LFAST modules work together as a single unit called Zipwire. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules.

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

46.2 Introduction

Zipwire is a group of modules that allow one MCU to have a fast, low pin count, serial communication link directly into the memory mapped peripherals and/or memories of another MCU or smart ASIC. Zipwire is implemented in hardware so there is no CPU load for the intitiator or target node. Zipwire supports 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address at the target node.

Zipwire architecture is fully pipelined and support multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at 320 Mbaud using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate 20 MHz reference clock for a total of five pins.

Zipwire also supports a streaming mode for the transfer of large blocks of data.

In streaming mode Zipwire has a high transfer rate. For single random access, 32-bit read, from any 32-bit address location, the latency is approximately 1 μs.

46.3 Zipwire Block Diagram

Figure 46-1 shows two MCUs with Zipwire connected using a five wire interface. Both MCUs support Target and Initiator modes of Zipwire. The SIPI Bus Master Interface is used for all Target mode transactions. The SIPI Register Interface is used for all Initiator Transactions and initial setup. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST reference clock coming from MCU B clock system. The Ref_Clock can come from either MCU. It is user configurable, but it must be the same clock that drives both LFAST modules.

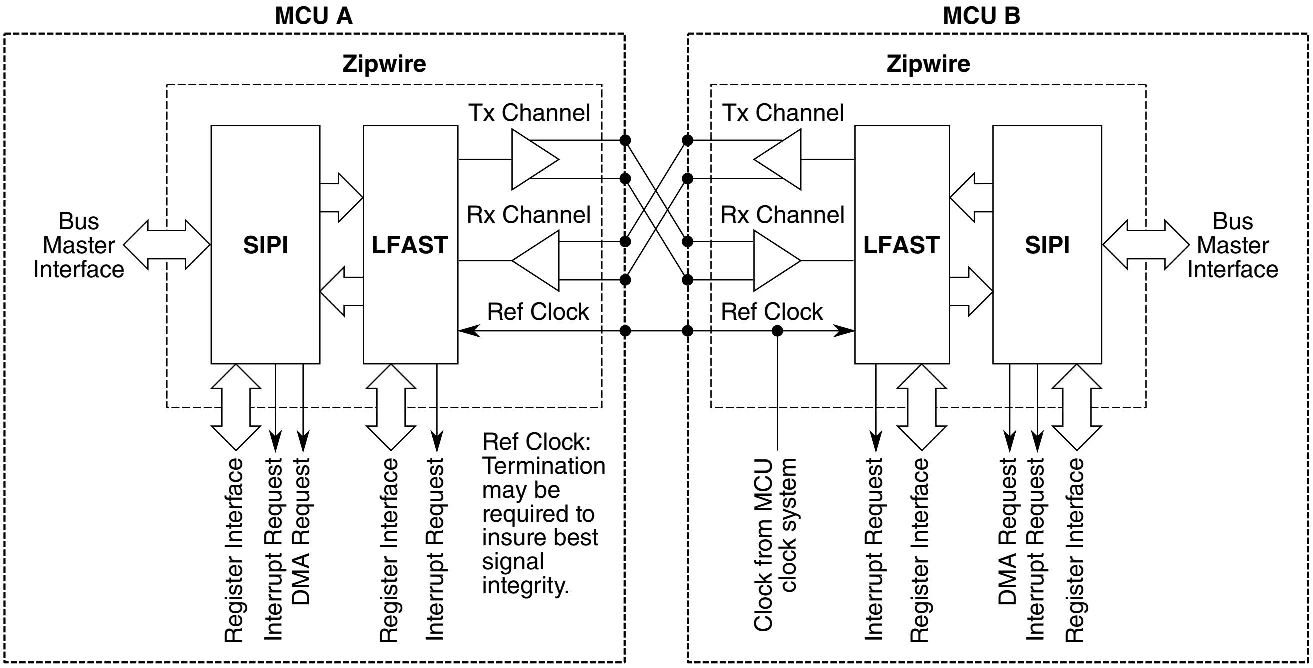


Figure 46-1. Zipwire connection diagram

46.4 Architecture

Zipwire is composed of several modules and system resources. The physical layer is LVDS with 1.2 V common mode voltage and 200mV swing.

The transport layer is a protocol called LFAST. LFAST is an asynchronous prototcol, using non-return to zero encoding. The LFAST protocol is composed of the following:

- A fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the incoming data.

- Followed by an 8-bit LFAST header, that defines the channel number and the size of the LFAST payload.
- Finally the payload, which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI). SIPI runs on top of LFAST and is fully encapsulated within the LFAST payload. SIPI uses four fixed sizes of LFAST payload in Zipwire:

- 32-bit
- 64-bit
- 96-bit
- 288-bit

The SIPI protocol implements a suite of commands initiated by one MCU, for reading and writing any 32-bit address location in another connected MCU. The SIPI protocol allows either or both MCUs to initiate commands. The protocol also supports interrupt requests from one MCU to the other.

The SIPI module implements both the initiator part of the protocol and the recipient part of the protocol. The module implements four SIPI Initiator channels that can run independently of each other and can run simultaneously. The SIPI module is a bus master on the low speed XBAR. As the target MCU for a command from the inititating MCU, SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

The initiator part of the module is connected to the DMA controller and is able to generate DMA requests as a command is completed. This allows a series of read or write commands to be queued in the Inititator MCU and executed by DMA and SIPI without CPU intervention at either Initiator or Target MCU.

46.5 Zipwire interconnections

LFAST has the following connections:

- Tx and Rx configuration is controlled by LFAST Control registers.
- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers.

- Tx Data Port/Rx Data Port — Directly connected to the SIPI module. Allows received data to be efficiently transferred to SIPI and transmit data to be transferred from SIPI.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers and SIPI Interface Registers.
- DMA connections — Allows SIPI command sequences to be queues and initiated without CPU intervention.
- Crossbar master port — Allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port — Directly connected to the LFAST module. Allows received data to be efficiently transferred from LFAST and transmit data to be transferred to LFAST.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

46.6 Zipwire performance

Two aspects of performance are considered:

- Bandwidth — The rate at which data can be read or written between two nodes. It assumes that read or write commands from the Inititator node, can be sent continuously at the highest speed the Target node can consume those commands.
- Latency — The time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

46.6.1 Read performance

A Zipwire read operation consists of three stages:

- Inititator sends SIPI Read command to Target.

- Target parses the received command and runs a master bus cycle to read the data.
- Target sends the SIPI Read response back to the Initiator.

A SIPI Read command consists of:

- 16-bit header
- 32-bit read address
- 16-bit CRC
- 64 bits total

A SIPI Read response consists of:

- 16-bit header
- 32-bit read data
- 16-bit CRC
- 64 bits total

The SIPI message is encapsulated in a LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 64-bit payload
- 1-bit stop bit
- 89 bits total

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 ns

Therefore, an 89-bit LFAST transmission of a SIPI 64-bit Read command or 64-bit Read response takes 278 ns (= 89 × 3.13 ns).

The SIPI module takes thirteen system clock cycles to parse a command and create a Read response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being read.

Assumptions:

- Average of 6 × XBAR cycles to read different memories.
- XBAR clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, time for SIPI to read an address location is:

(13 × 50 MHz clocks) + (6 × 100 MHz clocks) = 320 ns

46.6.1.1 Read bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Read response.

The time to transmit a command or response is less than the time to process the command. Therefore, the bandwidth is limited by the time to process the message.

Read Bandwidth = 4 bytes in 320 ns = 12.5 MB/second

46.6.1.2 Read latency

The latency is the time taken to send a Read command, process the command, and send a Read response.

Read Latency = 278 ns + 320 ns + 278 ns = 876 ns

46.6.2 Write performance

A Zipwire write operation consists of three stages:

- Initiator sends SIPI Write command to Target.
- Target parses the received command and runs a master bus cycle to write the data.
- Target sends the SIPI Write response back to the Initiator.

A SIPI Write command consists of:

- 16-bit header
- 32-bit write address

- 32-bit write data
- 16-bit CRC
- 96 bits total

A SIPI Write acknowledge consists of:

- 16-bit header
- 16-bit CRC
- 32 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 96-bit payload (command) or 32-bit (acknowledge)
- 1-bit stop bit
- 121 bits total (command) or 57 bits (acknowledge)

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 ns

Therefore, the time for an LFAST transmission is:

- 121-bit LFAST transmission of a SIPI 96-bit Write Command takes 378 ns
- 57-bit LFAST transmission of a SIPI 32-bit Write Acknowledge takes 178 ns

The SIPI module takes thirteen system clock cycles to parse a command and create a Write response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being written.

Assumptions:

- Average of 5 × XBAR cycles to write different memories.
- XBAR clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, the time for SIPI to write an address location is:

(13 × 50 MHz clocks) + (5 × 100 MHz clocks) = 310 ns

46.6.2.1 Write bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Write Acknowledge response.

The time to transmit the command is longer than the time to process the command, or the time to transmit the Write Acknowledge. Therefore, the bandwidth is limited by the time to transmit the command.

Write Bandwidth = 4 bytes in 378 ns = 10.6 MB/second

46.6.2.2 Write latency

The latency is the time taken to send a Write command, process the command, and send a Write Acknowledge.

Write Latency = 378 ns + 310 ns + 178 ns = 866 ns

Chapter 47
Serial Interprocessor Interface (SIPI)

47.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Serial Interprocessor Interface (SIPI) is an application layer protocol which runs on top of the LFAST (LVDS Fast Asynchronous Serial Transmission) module. It is used by the local device to access the shared memory of a remote device. SIPI defines point-to-point full duplex communication between two devices, where LFAST works as a physical medium of communication between both the devices.

47.1.1 Scalability

The SIPI protocol is designed to provide a sophisticated, high bandwidth, multimaster, multi-channel memory interface between 2 devices using few interconnecting signals. But the protocol is designed in such a way that a subset of the protocol can be implemented, where die area is more important than features.

Main scalable features:

- Number of concurrent channels:
  - Full Implementation = 4
  - Minimum Implementation = 1
- Full implementation has a node both as Initiator and Target of commands. Minimum implementation either Initiator or Target.
- Full Implementation includes a block transfer feature, but this feature is optional.

The rest of this section describes a full implementation of SIPI which includes:

- Advanced High-Performance Bus (AHB-Lite) master interface
- Direct Memory Access (DMA) interface
- LFAST Tx/Rx (transmit/receive) interfaces along with Peripheral Bus Interface (IPS)

47.2 Overview

An instance of SIPI can act as initiator, or target or both. SIPI can access the shared memory directly through its AHB master interface or through its DMA interface. DMA interface is used when the node acts as an initiator while the AHB Master interface will be used when the node acts as target. SIPI has four channels, with one channel (Channel 2) having data streaming capability. Payload width for channel 0, 1 and 3 is 32 bits, and for channel 2 data widths can be 32 bits, or 256 bits when streaming. Any of these channels can be used for DMA access or bus interface access depending on the setting of the SIPI_CCR*n*. CRC encoder calculates the CRC on the command frame. Then the SIPI initiator appends the CRC to the end of the frame before transmission.

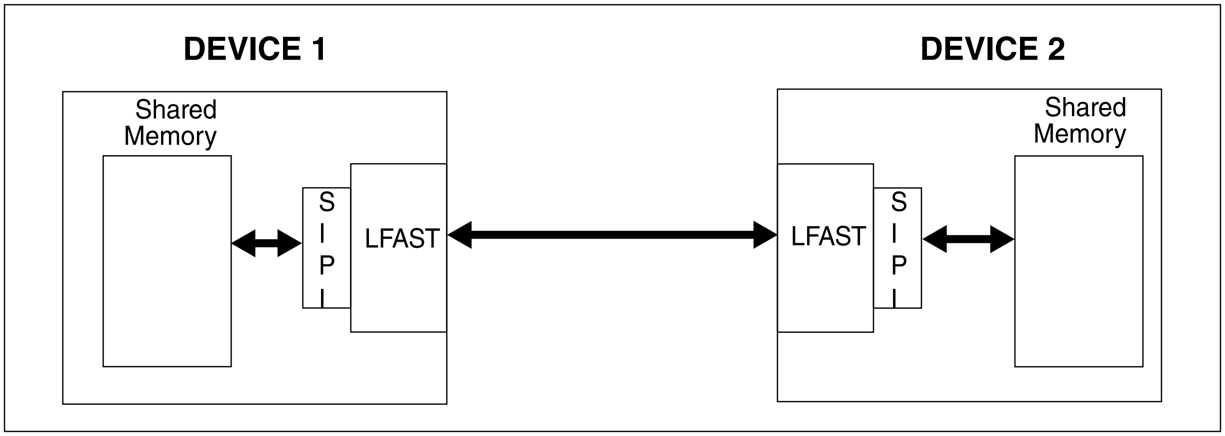


Figure 47-1. Interprocessor communication diagram

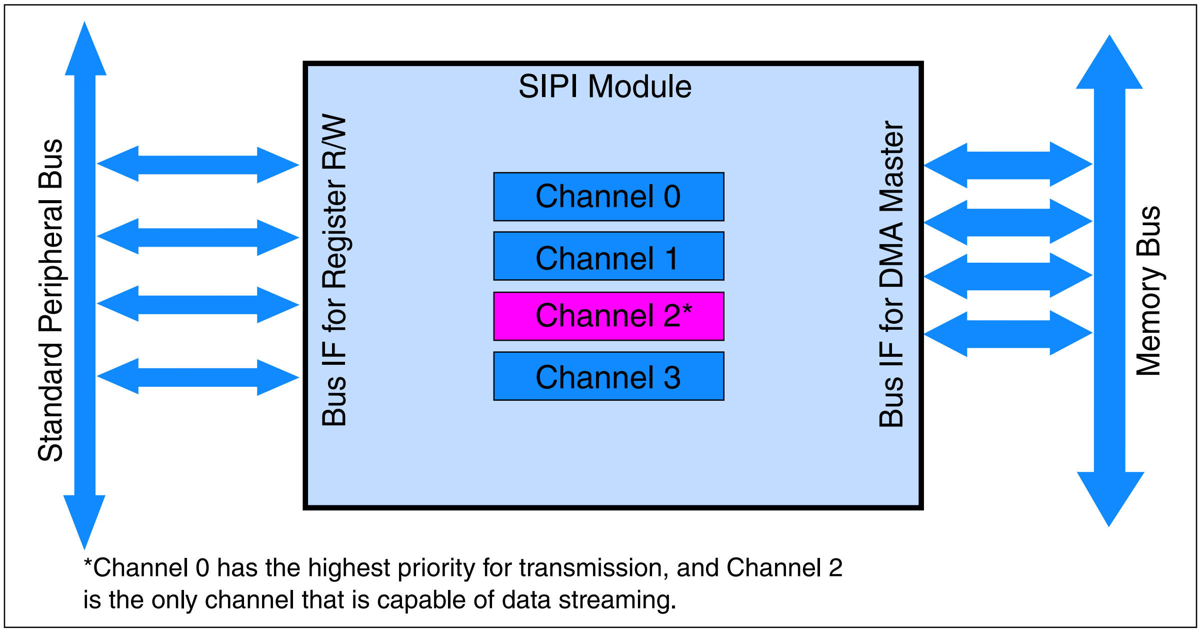
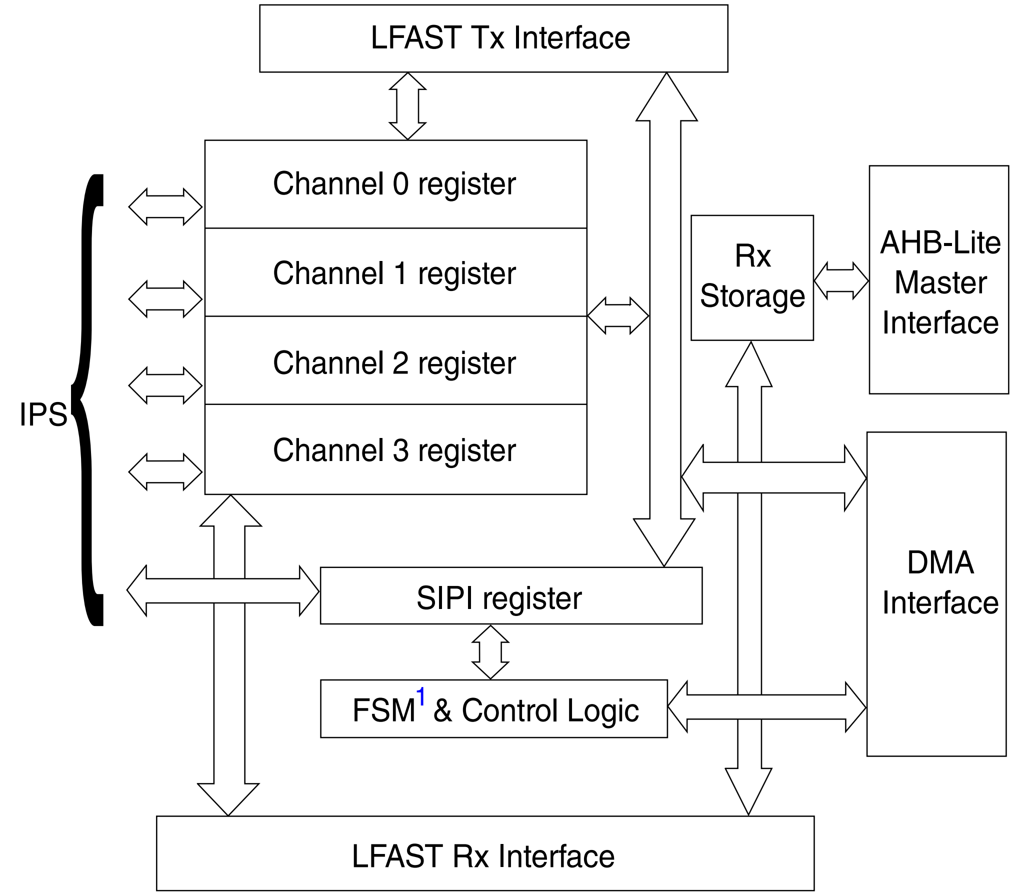


Figure 47-2. SIPI module

47.3 SIPI block diagram



[1] Finite State Machine

Figure 47-3. SIPI block diagram

47.4 Feature description

This section describes the features of the SIPI module.

47.4.1 Main features

- Point-to-point communication between two devices
- Full duplex communication
- Four channels, including one channel with data streaming capability

- Configurable DMA access for each channel
- CRC protection mechanism
- Timeout protection mechanism
- Fixed priority channel selection
- Data size up to 256 bits on streaming channel
- Common tag pool for assigning sequential transfer IDs to every new transfer
- AHB master interface which is used by target node to access shared memory
- Target node contains a set of nine 32-bit internal registers to store commands
- Up to two outstanding requests are supported at initiator

47.4.2 Standard features

- IPS bus interface (PBRIDGE)
- SPP DMA2x bus interface
- AHB Master Interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

47.5 SIPI operation from reset

When the SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

47.6 Functional description

47.6.1 External signals

The SIPI has no chip external signals.

47.6.2 Frame format

All frames have the same general format:

- 16 bit header
- Address, Address and Data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID. The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

47.6.2.1 Register read request

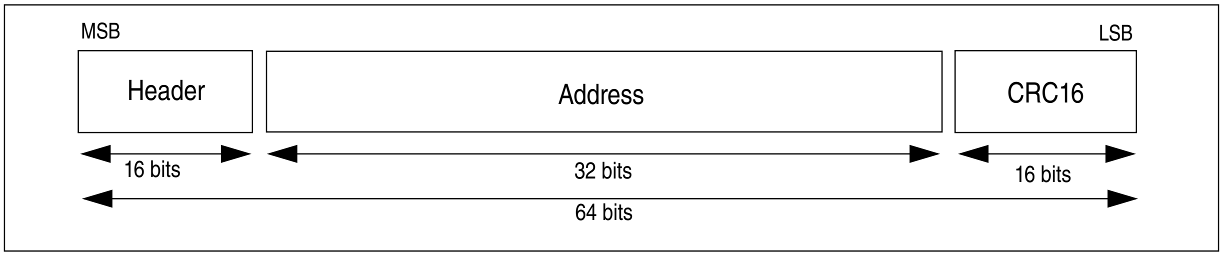


Figure 47-4. SIPI register read request

47.6.2.2 Register read response, ID request response

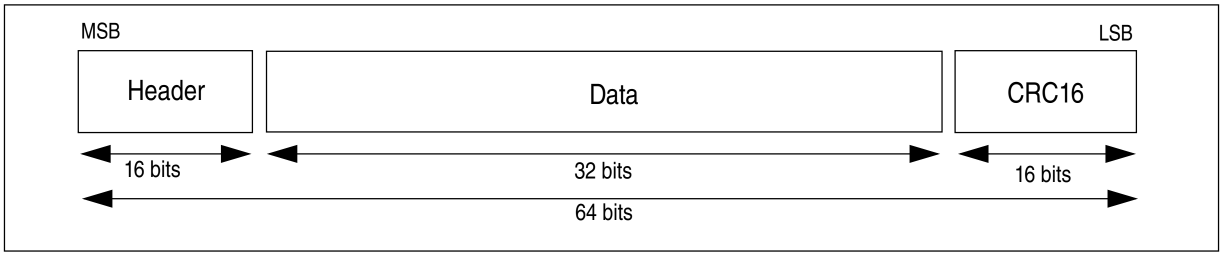


Figure 47-5. SIPI read response

47.6.2.3 Register write request

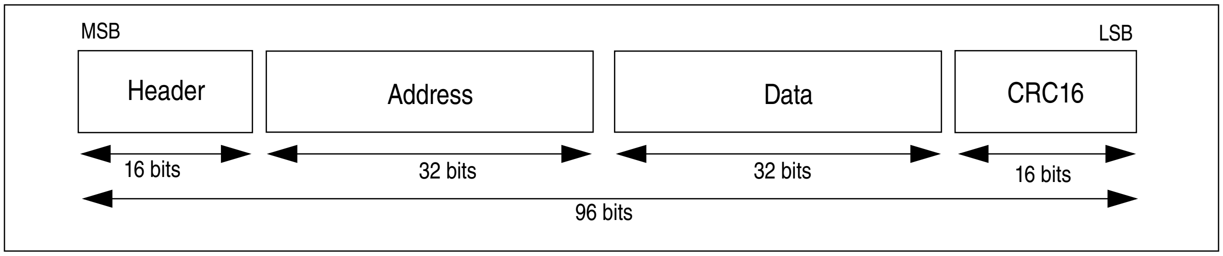


Figure 47-6. SIPI register write request

47.6.2.4 Trigger transfer, ID transfer, write acknowledge and streaming write acknowledge

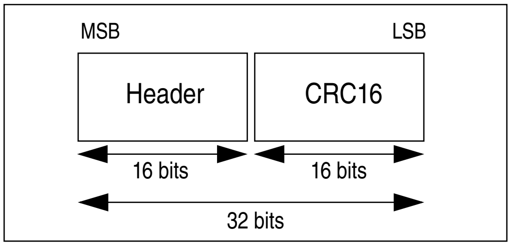


Figure 47-7. SIPI write acknowledge

47.6.2.5 Streaming write request

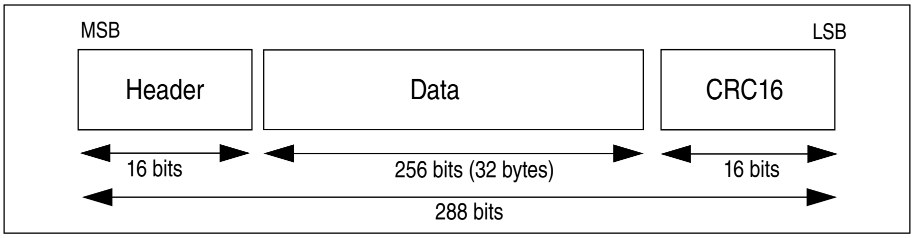


Figure 47-8. Streaming write request format

NOTE

Direct write operations are used to set the streaming address.

Streaming data write is performed using the format shown in Figure 47-8.

47.6.2.6 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

47.6.2.6.1 SIPI header coding

Figure 47-9, Table 47-1, Figure 47-10, and Table 47-2 show how the SIPI header bits are coded.

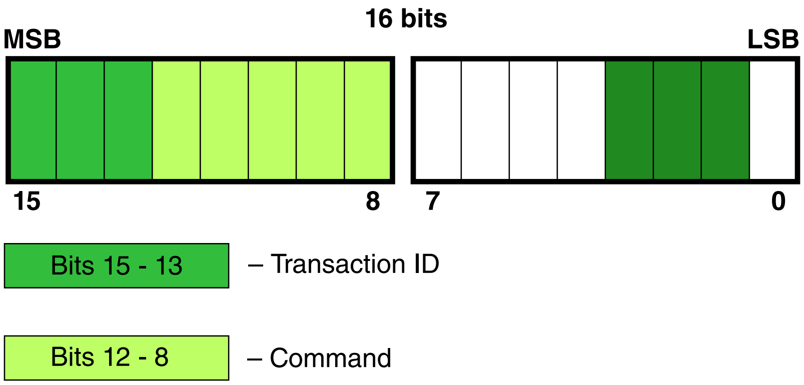


Figure 47-9. SIPI header coding

Table 47-1 shows the command coding for the bits[12:8] of the header.

Table 47-1. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
00	Read 8 bits	64
01	Read 16 bits	64
02	Read 32 Bits	64
03	Reserved	—
04	Write 8 bits with ACK	96
05	Write 16 bits with ACK	96
06	Write 32 bits with ACK	96
07	Reserved	—
08	ACK – OK	32
09	ACK – Fault	32
0A	Read Answer – OK	64
0B	Reserved	—
0C	Trigger comm and with ACK	32
0D	Reserved	—
…	…	…
11	Reserved	—
12	ID Register Read Request	32
13	Reserved	—
…	…	…
16	Reserved	—
17	Stream Data with ACK (32 bytes)	288
18	Reserved	—
…	…	…
1F	Reserved	—

Figure 47-10 shows the channel number field in the header and Table 47-2 shows the channel number coding.

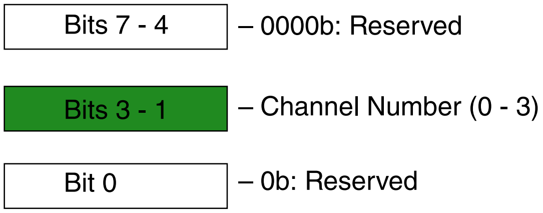


Figure 47-10. SIPI header channel field

Table 47-2. SIPI header channel number coding

SIPI channel name	Channel number coding in header(s)		Comment
	SIPI channel coding select field (SIPI_MCR[CHNSB])		
	Code table I (SIPI_MCR[CHNSB] = 1)	Code table II (SIPI_MCR[CHNSB] = 0)	
0 (Channel A)	000b	100b	In use
1 (Channel B)	001b	101b	In use
2 (Channel C)	010b	110b	In use
3 (Channel D)	011b	111b	In use
4 (Channel E)	100b	000b	Reserved
5 (Channel F)	101b	001b	Reserved
6 (Channel G)	110b	010b	Reserved
7 (Channel H)	111b	011b	Reserved

47.6.2.7 Address field

The address field is 32 bits wide with the MSB transmitted first.

47.6.2.8 Payload field

Table 47-3 shows the payload sizes of LFAST frames.

Table 47-3. Payload size of LFAST channel frame

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16

Table continues on the next page...

Table 47-3. Payload size of LFAST channel frame (continued)

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
101b	101b	256	32
110b	110b	512	64
111b	111b	288	36

Table 47-4 shows the converted coding of LFAST for a given SIPI code.

Table 47-4. Converted coding of LFAST channel code for SIPI headers

LFAST Code	SIPI Code	Channel (hex)[1]
0100b	100b	A
0101b	101b	B
0110b	110b	C
0111b	111b	D
1000b	000b	E (not used by SIPI)
1001b	001b	F (not used by SIPI)
1010b	010b	G (not used by SIPI)
1011b	011b	H (not used by SIPI)

[1] SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

47.6.2.9 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). MSB is sent first in the data stream.

47.6.2.9.1 CRC field examples

47.6.2.9.1.1 Example 1 – 32 bit write on channel 1 with ID1

- Header = 260Ah
- Address = 1122_3344h
- Data = CCDD_EEFFh
- CRC = BF7Dh

47.6.2.9.1.2 Example 2 – 32 bit read on channel 2 with ID2

- Header = 420Ch
- Address = 89AB_CDEFh
- CRC = 6B80h

47.6.2.9.1.3 Example 3 – Event command on channel 3 with ID3

- Header = 6C0Eh
- CRC = B286h

47.7 Transfer types

This section describes the available transfer types of the SIPI module. The SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

47.7.1 Read transfer

A Read transfer can be of two types:

- Read Request Transfer (at initiator node)
- Read Response Transfer (at target node)

47.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in Figure 47-11.

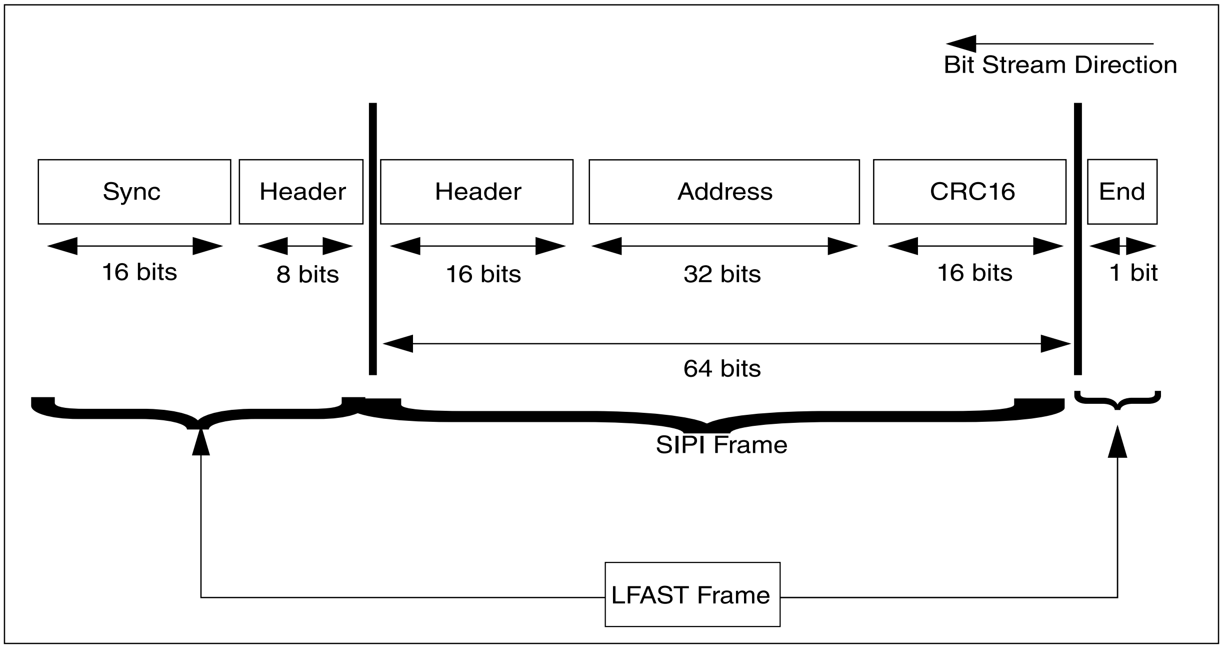


Figure 47-11. Read request transfer

47.7.2 Register read answer transfer

In response to a read request by the initiator node, the target node will send header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (see Figure 47-12). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (see Figure 47-13).

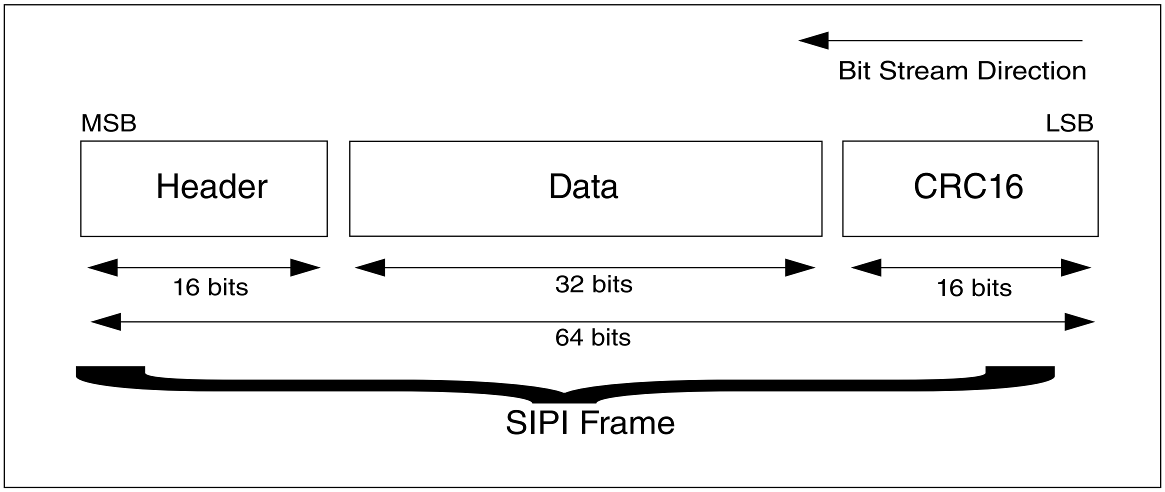


Figure 47-12. Read answer transfer

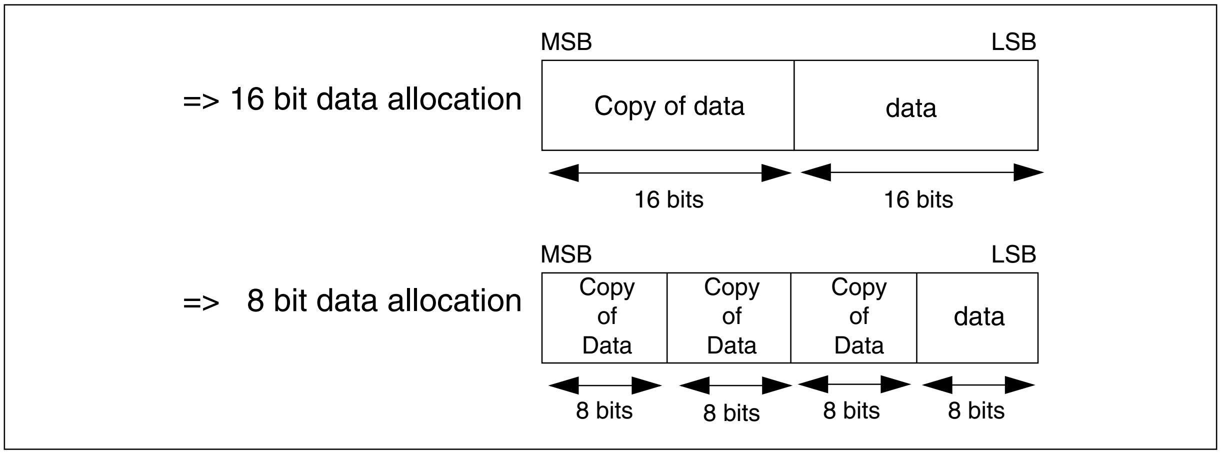


Figure 47-13. Data allocation

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as halfword enable.
  - Bit[0]:
    - 0 - halfword 1 enabled (MSB)
    - 1 - halfword 0 enabled (LSB)

47.7.3 Register Write transfer

Register Write transfer can be of two types:

- Normal write transfer - channels 0, 1, 2 and 3
- Streaming data transfer - channel 2 only

A Register Write transfer can be done through Normal Write transfer on channels 0, 1, 2 and 3 (see Normal write transfer) as shown in Figure 47-14.

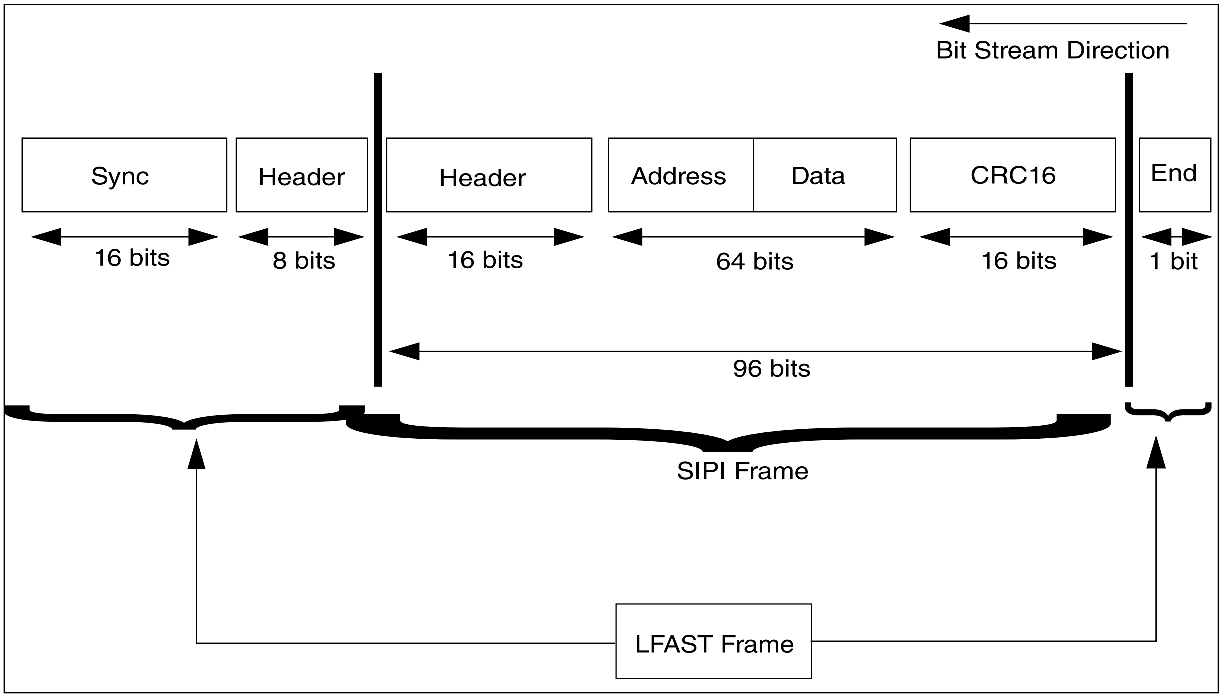


Figure 47-14. Register write transfer (showing LFAST frame encapsulation)

47.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and CRC as shown in Figure 47-15.

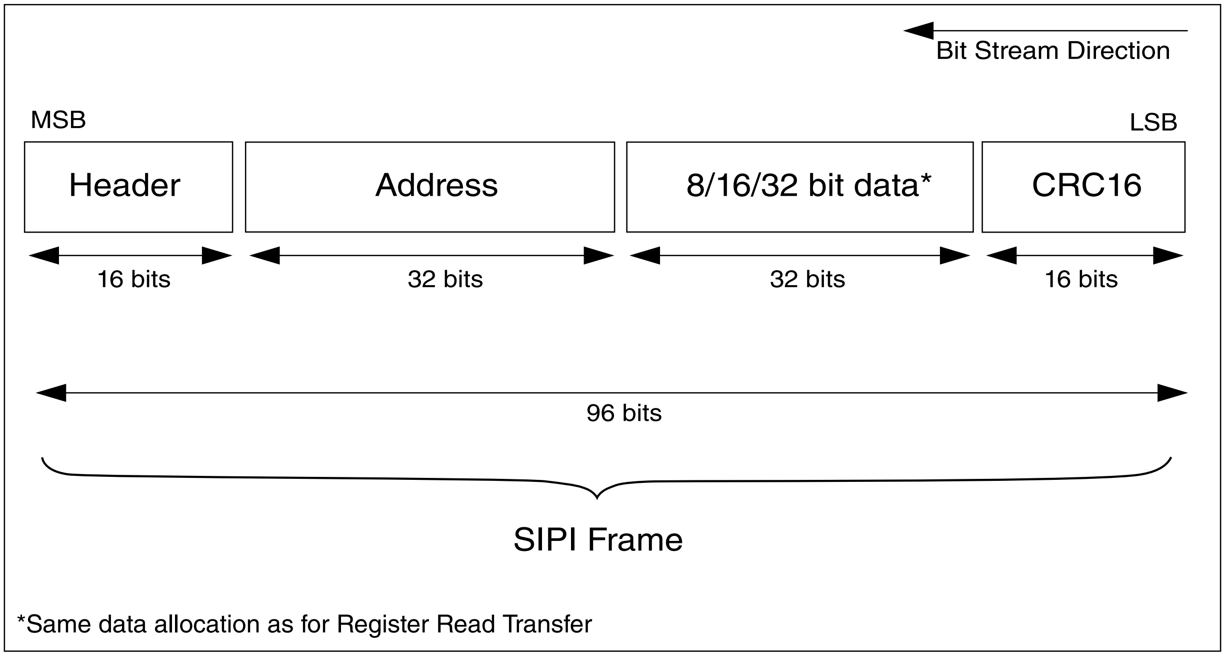


Figure 47-15. Write transfer (LFAST frame encapsulation is not shown)

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable.
  - Bit[0]:
    - 0 - half-word 1 enabled (MSB)
    - 1 - half-word 0 enabled (LSB)

47.7.3.2 Streaming write transfer

Streaming write transfer has 256 bits of payload.

47.7.3.2.1 Streaming write transfer with address

Setting the address is performed using a direct write transfer. The SIPI Maximum Count Register (SIPI_MAXCR) and SIPI Address Reload Register (SIPI_ARR) are written before the SIPI Address Count Register (SIPI_ACR). This is to avoid unwanted behavior of the SIPI attempting to access non-shared memory. This is only true for the very first streaming command, subsequent streaming command(s) may not need to write the SIPI_MAXCR and SIPI_ARR (see SIPI Max Count Register (SIPI_MAXCR), SIPI Address Count Register (SIPI_ACR) and SIPI Address Reload Register (SIPI_ARR)).

47.7.3.2.2 Streaming transfer with data

Figure 47-16 shows the packet structure of the streaming transfer containing data.

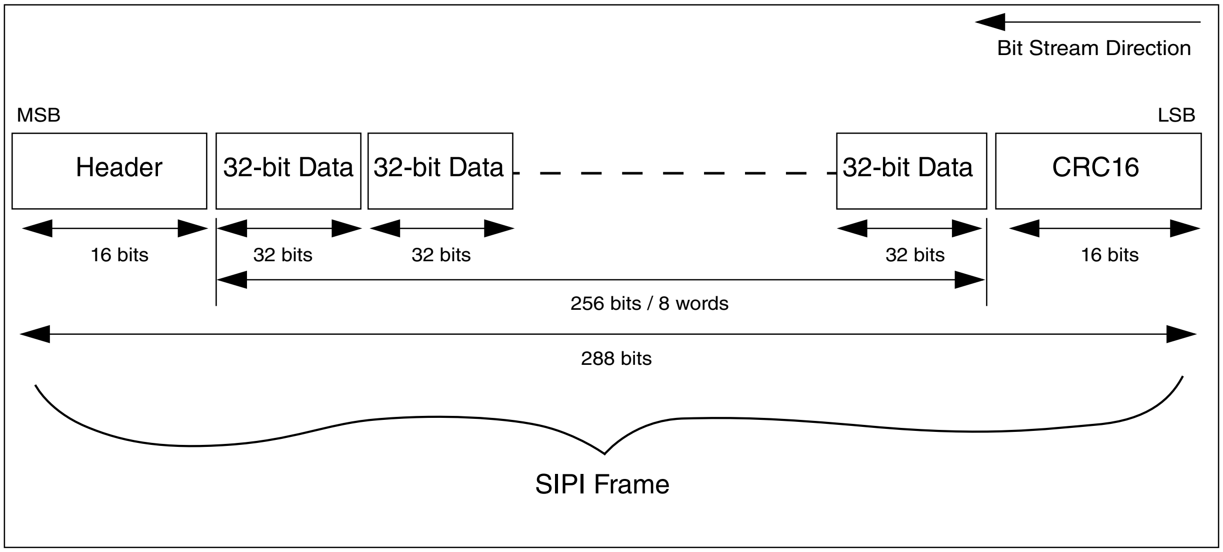


Figure 47-16. Streaming transfer with data

47.7.4 Write Acknowledge transfer

A Write Acknowledge transfer contains only header and CRC bits (see Write Aknowledge transfer. The CRC bits are calculated on the header field.

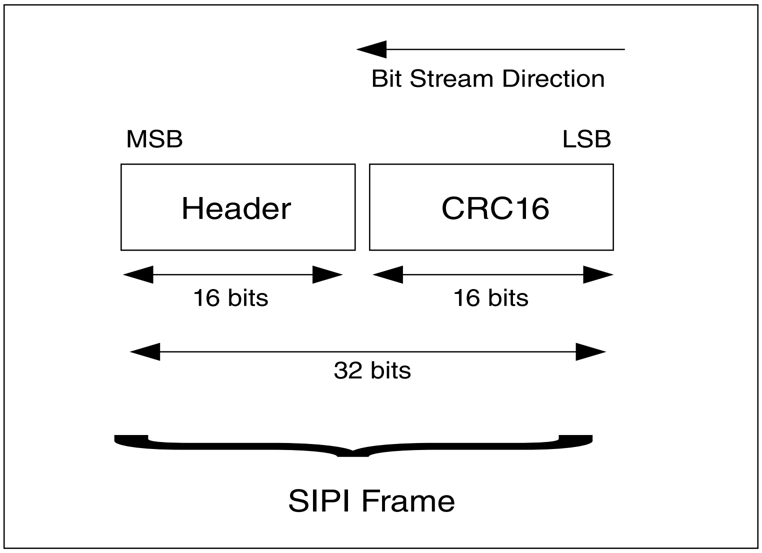


Figure 47-17. Write Aknowledge transfer (LFAST encapsulation is not shown)

47.7.5 ID request response

47.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in Figure 47-18.

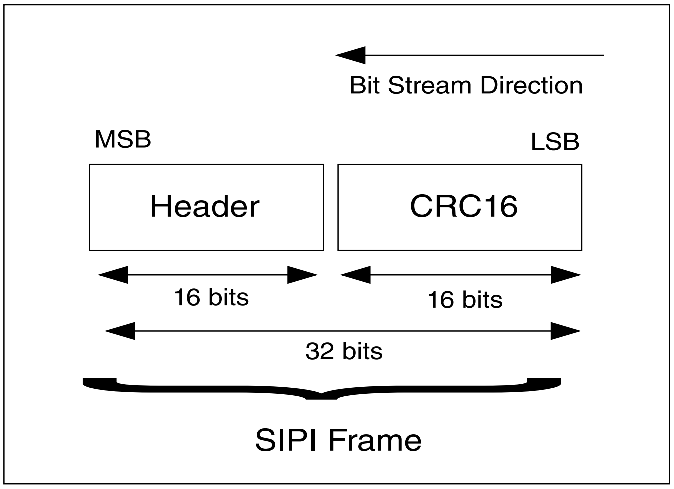


Figure 47-18. ID request transfer (LFAST encapsulation is not shown)

47.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in Figure 47-19.

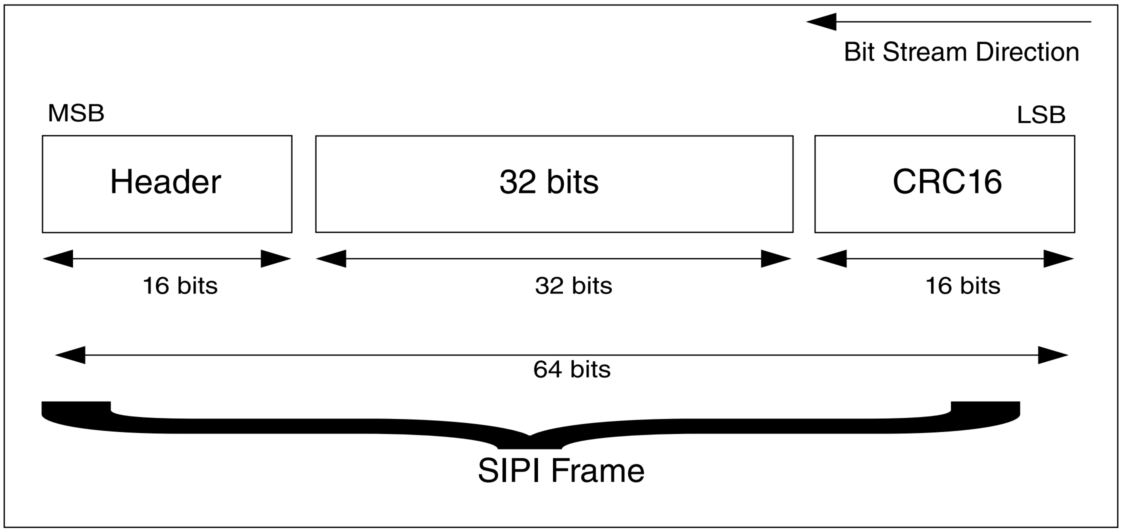
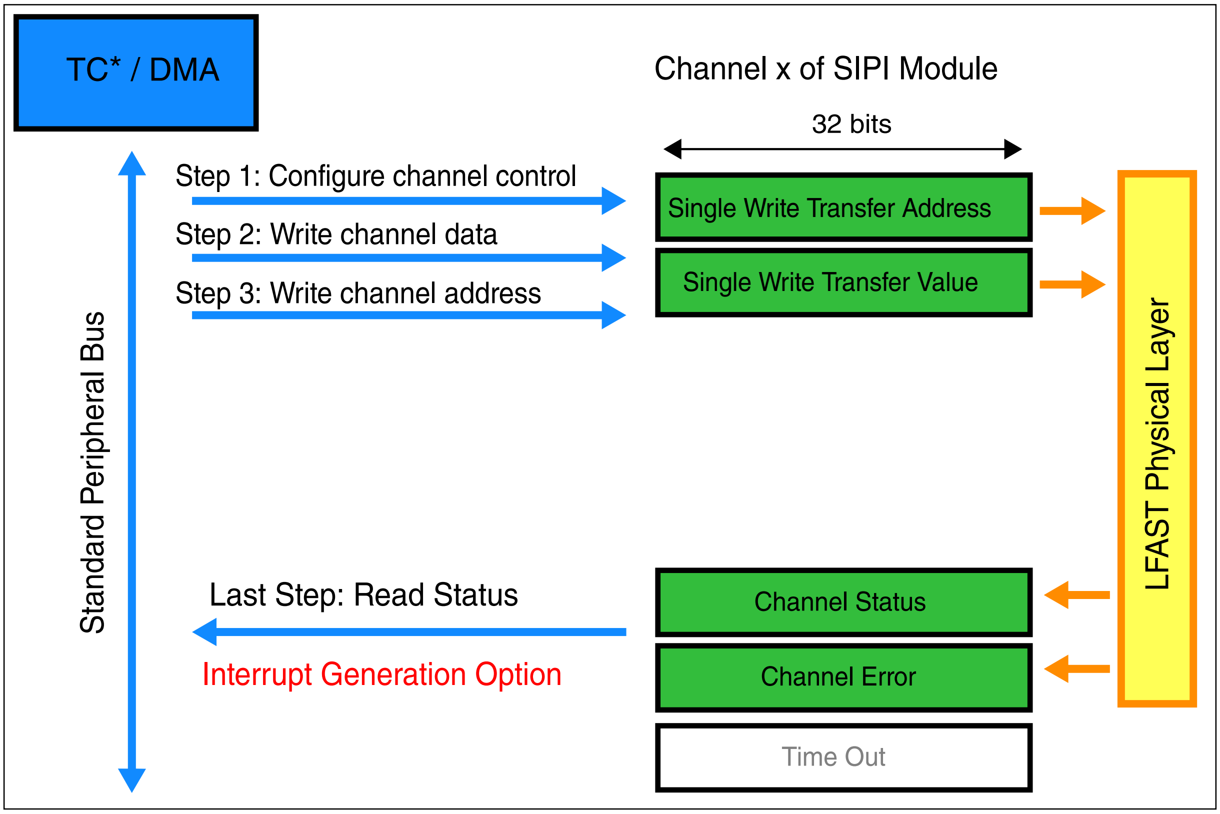


Figure 47-19. ID request response transfer

Note

The ID request response contains the value of the CHIP ID in place of data.

47.8 Transfer API and flow charts


TC* / DMA
Channel x of SIPI Module
32 bits
Step 1: Configure channel control
Step 2: Write channel data
Step 3: Write channel address
Single Write Transfer Address
Single Write Transfer Value
Standard Peripheral Bus
LFAST Physical Layer
Last Step: Read Status
Interrupt Generation Option
Channel Status
Channel Error
Time Out

*Note: TC = Transmit Command

Figure 47-20. SIPI single register write API

Implement the following steps to generate a single Write Transfer Request (Figure 47-20):

1. Configure data and SIPI_CCR*n* at the initiator node.
2. Configure SIPI_CAR*n* at the initiator node.

   As soon as the channel address register (SIPI_CAR*n*) is written and if CCR*n*[CHEN] = 1, the initiator SIPI will calculate the CRC on header, address and data field and will start transmitting data to LFAST.
3. Software polls the status register bits (SIPI_CSR*n*) to determine when the request completes, SIPI_CSR*n*[ACKR] = 1. An interrupt will be generated if the corresponding SIPI_CIR*n*[ACKIE] = 1.

On a single Write transfer request reception (Figure 47-21):

1. Target node will place the address, data and control information on its AHB master interface.
2. When the process in completed, the target node will generate an acknowledge frame and send it back to the LFAST.

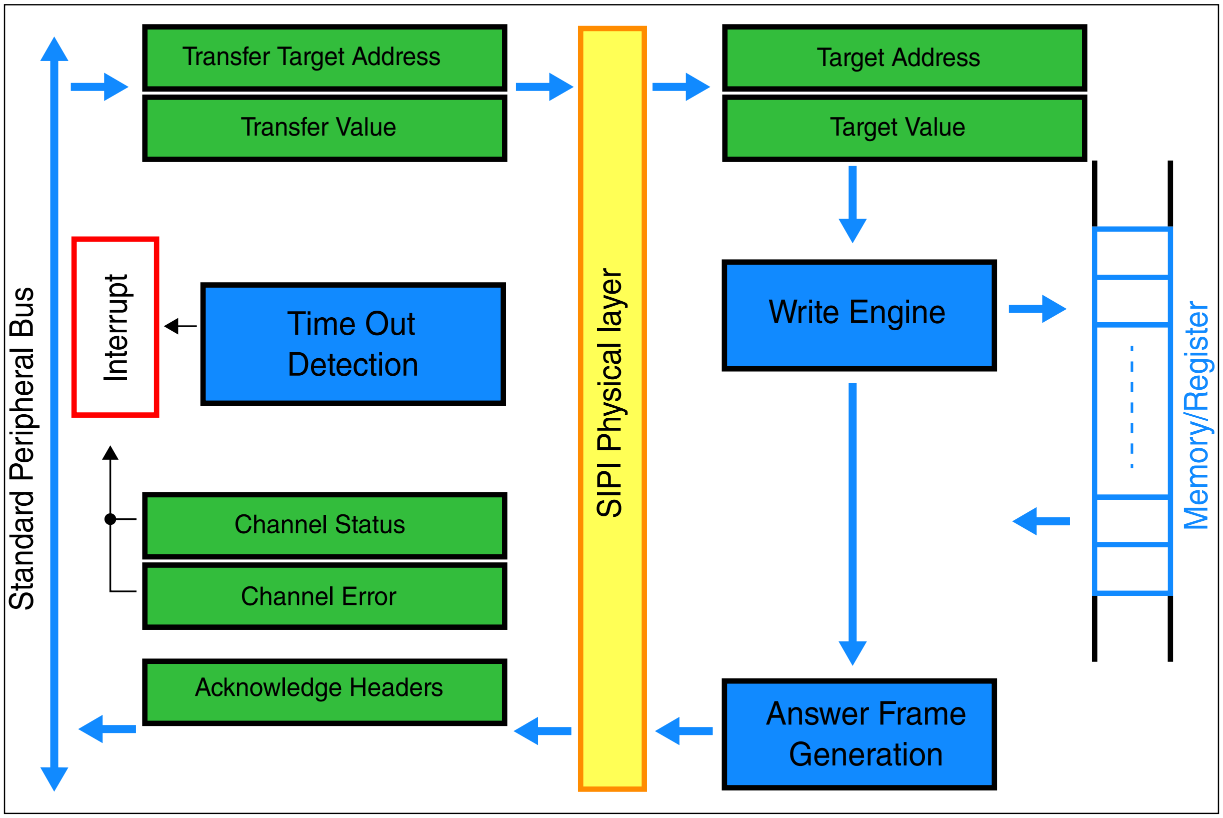


Figure 47-21. SIPI Single Register Write API – Flow Chart

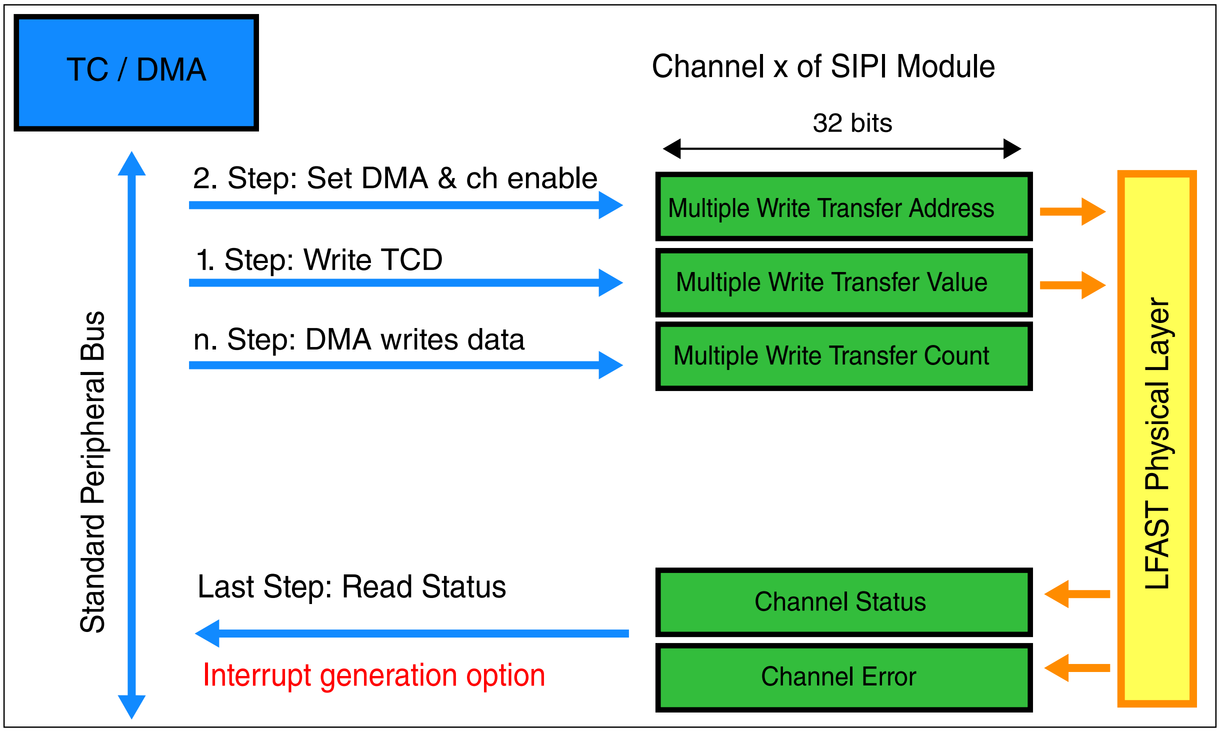


Figure 47-22. SIPI Multiple Register Write API

For multiple write transfer request generation (Figure 47-22):

1. Software will configure the Transfer Control Descriptor (TCD) of the DMA.
2. Software will write SIPI_CCR*n*[CHEN] = 1 and SIPI_CCR*n*[DAN] = 1.
3. SIPI will start copying data into the SIPI_CDR*n* through its DMA interface, depending on the transfer count and data registers size. SIPI_CDR2_0 should be written with the MSB.
4. When the copying process is completer, initiator SIPI will calculate CRC on header, address and data field and start transmitting data to LFAST.
5. Software should poll the SIPI_CSR*n* status register bits to determine if the request has completed. If SIPI_CSR*n*[ACKR] = 1, an interrupt will be generated (if the corresponding SIPI_CIR*n*[ACKIE] = 1).
6. If SIPI_CCR*n*[DEN] = 1, the SIPI request will transfer to the DMA controller. If not, the state machine go idle.
7. Steps 4–7 will be repeated.

On Multiple Write transfer request reception (Figure 47-22):

1. Target node will place the address, data and control information on its AHB Master Interface.
2. When the process in completed, target node will generate an acknowledge frame and send it back to the LFAST.

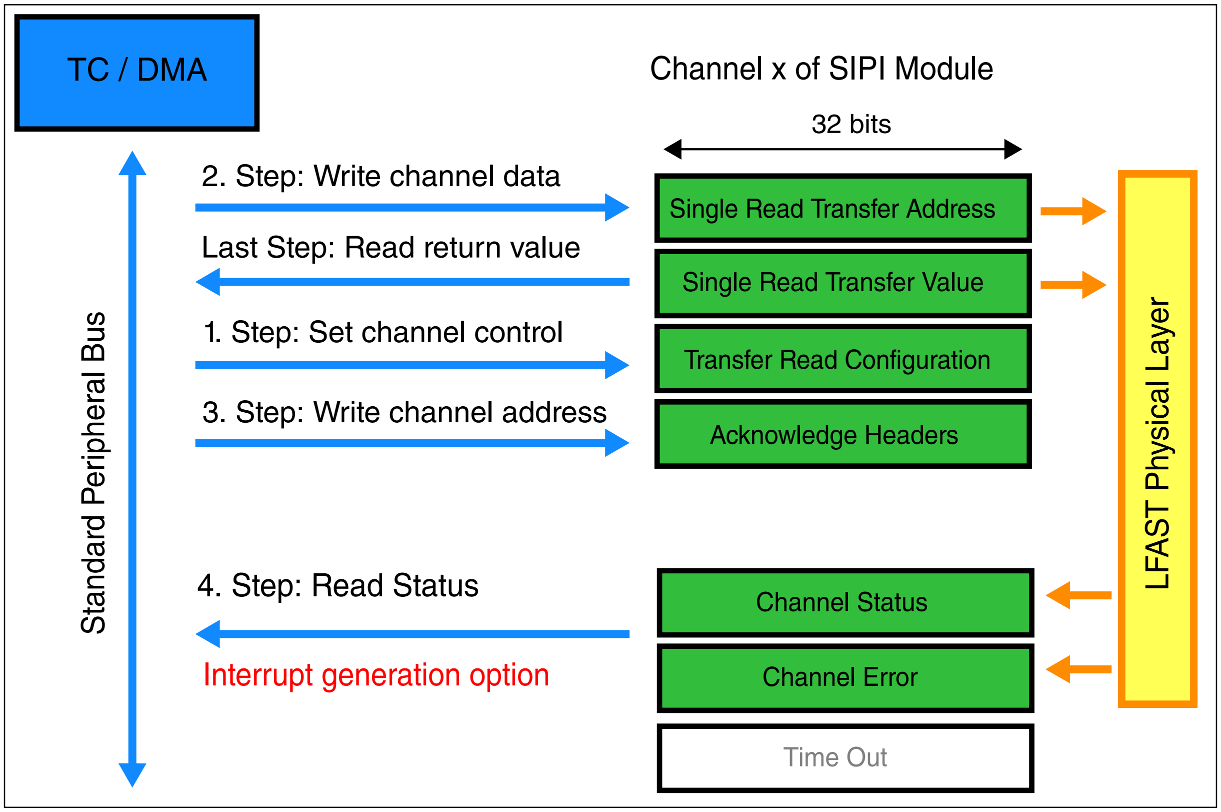

TC / DMA
Channel x of SIPI Module
32 bits
2. Step: Write channel data
Last Step: Read return value
1. Step: Set channel control
3. Step: Write channel address
4. Step: Read Status
Interrupt generation option
Standard Peripheral Bus
Single Read Transfer Address
Single Read Transfer Value
Transfer Read Configuration
Acknowledge Headers
Channel Status
Channel Error
Time Out
LFAST Physical Layer

Figure 47-23. SIPI Single Register Read API

For Single Read Transfer Request generation (Figure 47-23):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n*, with the last step by writing CAR*n*.
2. As soon as SIPI_CAR is written, the initiator SIPI will calculate CRC of the header and address fields and start transmitting data to the LFAST.
3. Software should poll CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1, an interrupt will be generated (if SIPI_CIR*n*[RAIE] = 1). If SIPI_CSR*n*[RAR] does not set then SIPI_ERR[TOE*n*] = 1 which indicates a timeout has occured.
4. If SIPI_CSR*n*[RAR] = 1 then software can read the data register, or can other necessary action if SIPI_CSR*n*[RAR] = 0.

On Single Read transfer request reception (Figure 47-23):

1. Target node will place the address and control information on its AHB Master Interface.
2. When the process in completed, target node will send read response back to LFAST.

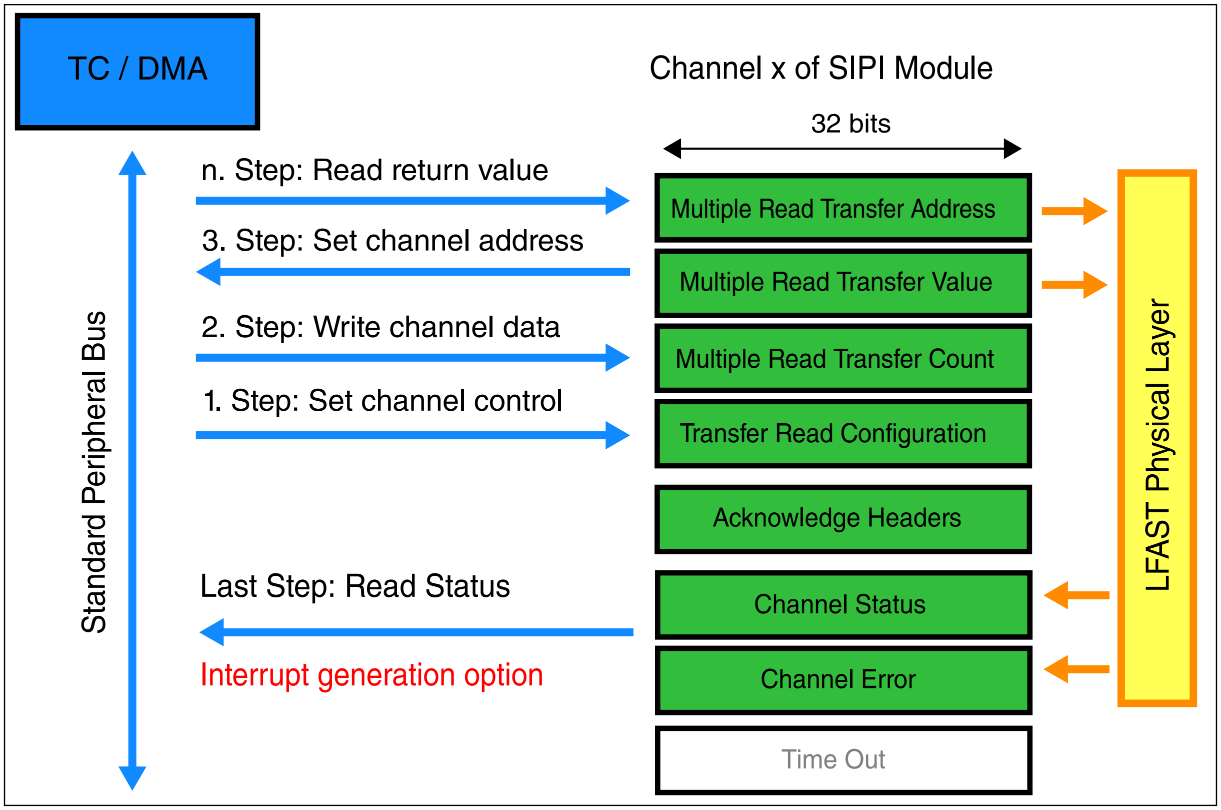

TC / DMA
Channel x of SIPI Module
32 bits
Standard Peripheral Bus
n. Step: Read return value
3. Step: Set channel address
2. Step: Write channel data
1. Step: Set channel control
Last Step: Read Status
Interrupt generation option
Multiple Read Transfer Address
Multiple Read Transfer Value
Multiple Read Transfer Count
Transfer Read Configuration
Acknowledge Headers
Channel Status
Channel Error
Time Out
LFAST Physical Layer

Figure 47-24. SIPI multiple register read API

For multiple read transfer request generation (Figure 47-24):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n*, with the last step by writing SIPI_CAR*n*.
2. As soon as channel address register is written, initiator SIPI will calculate CRC on the header and address fields, then starts transmitting data to the LFAST.
3. Software should poll SIPI_CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1 and SIPI_CIR*n*[RAIE] = 1 an interrupt will be generated. If SIPI_CSR*n*[RAR] does not set, then SIPI_ERR[TOE*n*] = 1 which indicates a timeout.
4. Steps 2–3 will be repeated for multiple requests.

On Multiple Read transfer request reception (Figure 47-24):

1. Target node then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

Figure 47-25, Figure 47-26, Figure 47-27 and Figure 47-28 show the SIPI data streaming of data.

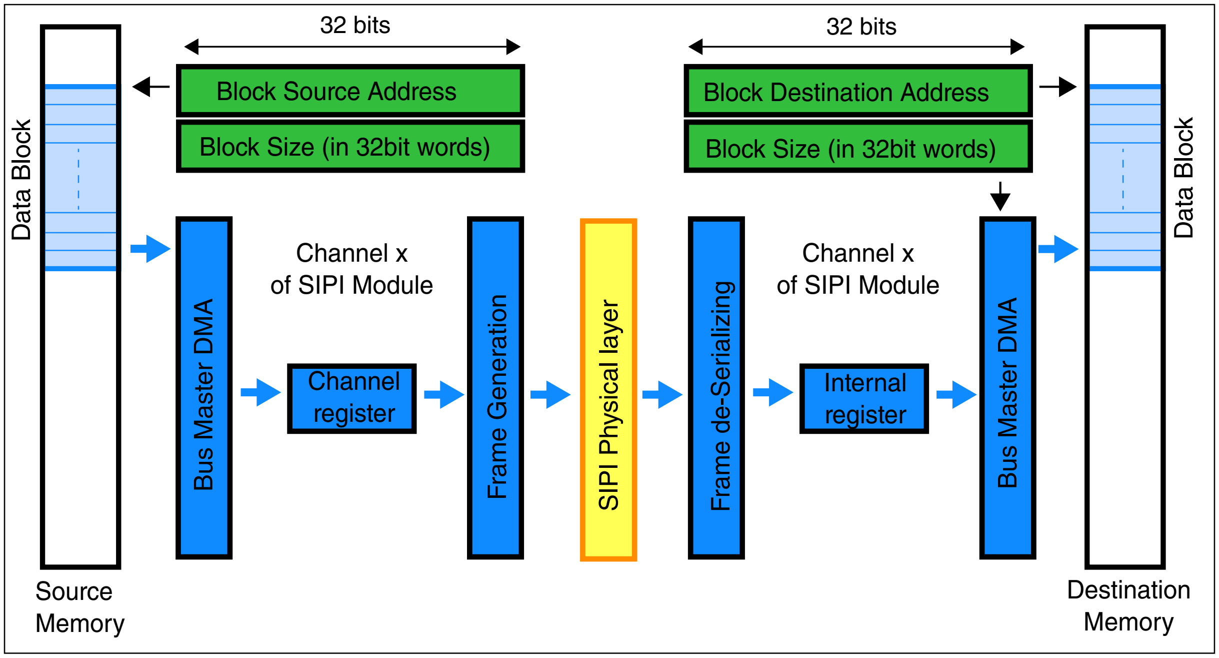


Figure 47-25. SIPI data stream model

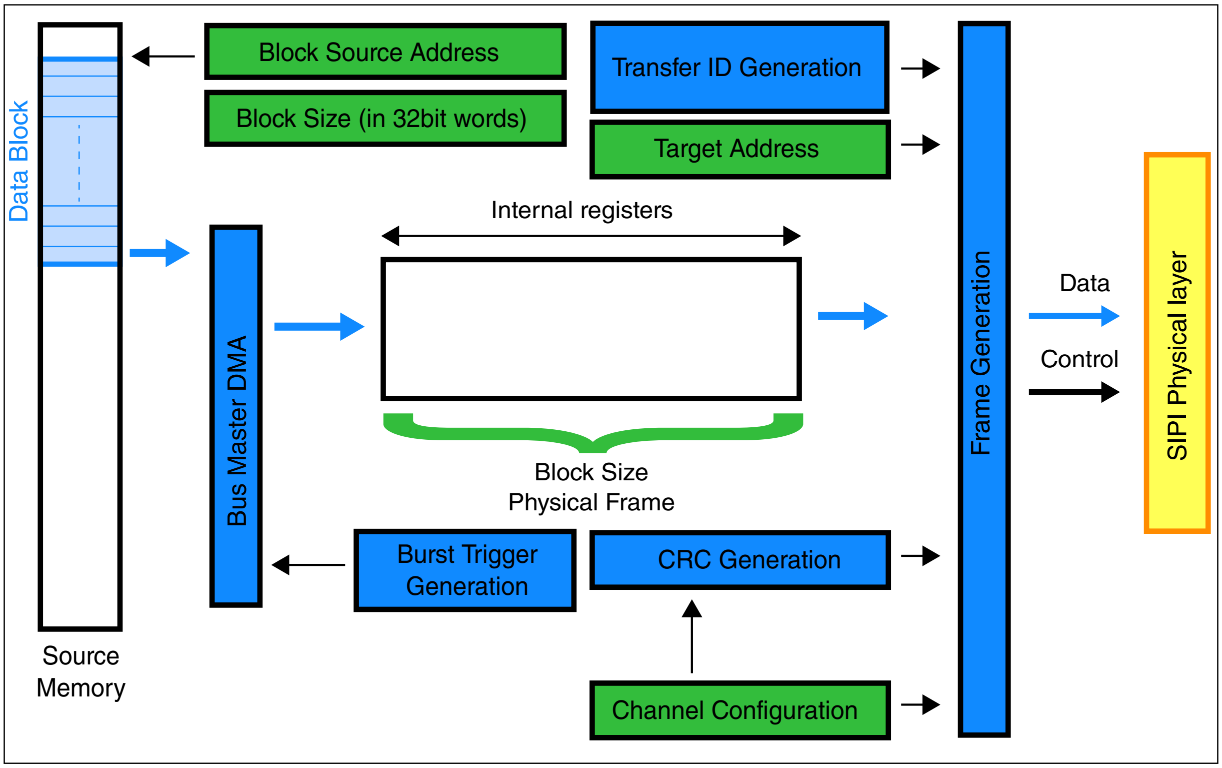


Figure 47-26. SIPI data stream model - Initiator

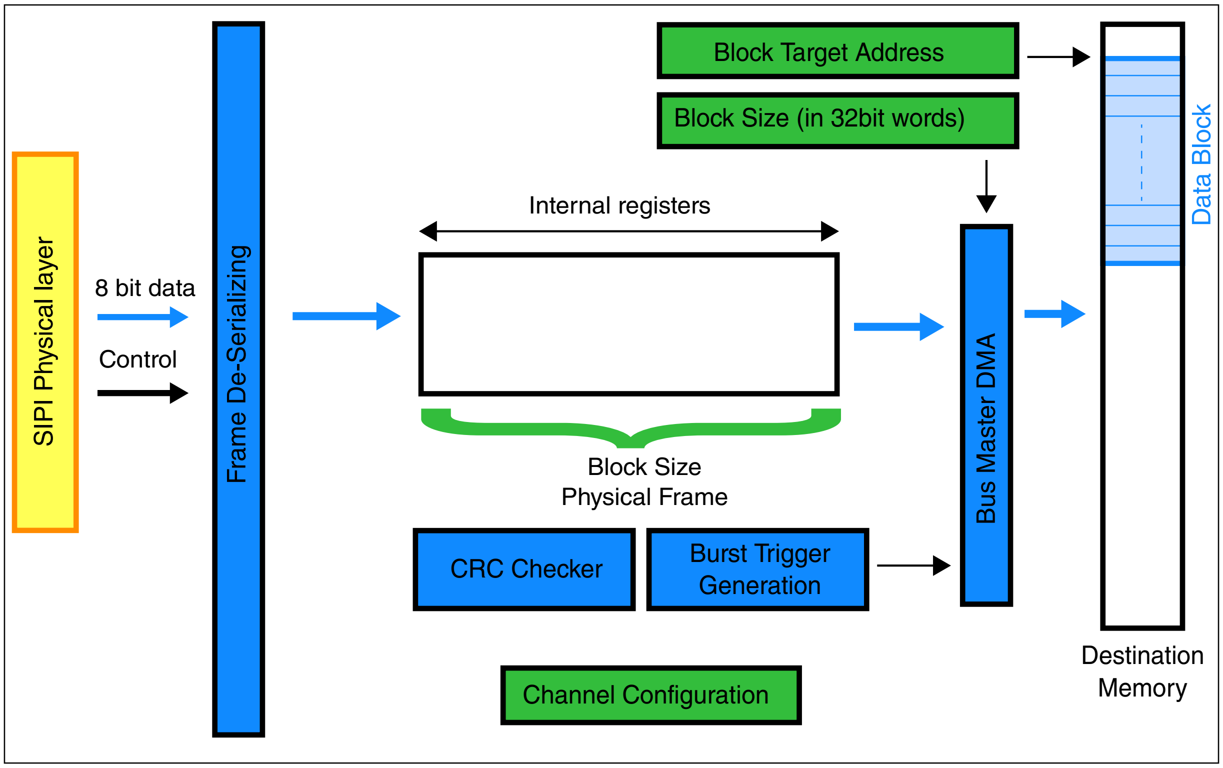


Figure 47-27. SIPI data stream model - Target

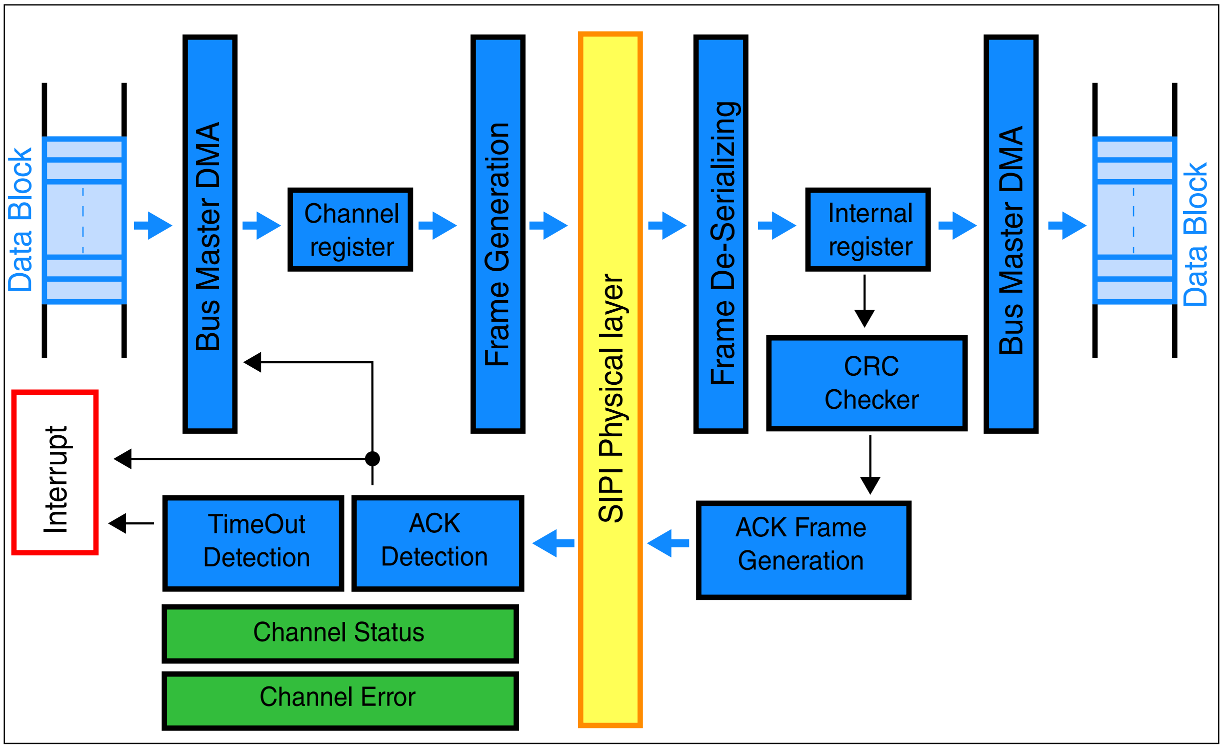


Figure 47-28. SIPI data stream with acknowledge trigger

47.9 DMA programming sequence

The DMA programming sequence is as follows:

1. Software configures TCD (DMA data).
2. DMA transfers from RAM to SIPI registers.
3. SIPI starts transferring the data through LFAST Tx ports.
4. If an acknowledge or fault response is received, SIPI will repeat the process. If any error response is received, SIPI will move to an idle state. An interrupt will be flagged by SIPI, and software will take control of the SIPI.
5. If SIPI has received a read response, and error flags are not set on the LFAST Rx port, SIPI will assert ipd_request. In case of error, move to Step 8.
6. As soon as ipd_request is asserted, DMA transfer starts between the SIPI registers and RAM.
7. If DMA transfer minor loop/major loop is complete, and the request is negated, ipd_request is negated.
8. If an error occurs:

- The SIPI generates an interrupt
- Software takes control of the SIPI

9. Steps 5 – 8 are repeated until the transfer is finished.

47.10 Modes of operation

SIPI has three operating modes:

- Initialization
- Normal
- Module Disable

After hardware reset the SIPI is in module disable mode, which helps reduce power consumption.

47.10.1 Initialization mode

To enter Initialization mode software writes SIPI_MCR[INIT]=1 SIPI_MCR[MOEN] must be set before attempting to write SIPI_MCR[INIT]). To exit Initialization mode software writes SIPI_MCR[INIT]=0 (see SIPI Module Configuration Register (SIPI_MCR)).

All message transfers are stopped when in Initialization mode. If software invokes Initialization mode when a bus transfer is in progress, the transfer will be aborted immediately even if the transfer has not completed.

Note

It is recommended that software checks the state of SIPI (SIPI_MCR[MOEN]) before setting SIPI_MCR[INIT].

47.10.2 Normal mode

Once software has completed initialization of SIPI (SIPI_MCR[INIT]=1), it can enter Normal mode by writing SIPI_MCR[INIT]=0. SIPI needs to be in Normal mode for all data transfers (see SIPI Module Configuration Register (SIPI_MCR)).

Note

SIPI must be enabled before a attempting to write SIPI_MCR[INIT] (SIPI_MCR[MOEN]=1).

47.10.3 Module Disable (MD)

MD mode in the SIPI is used to help reduce power consumption. By default, SIPI is in Disable mode, SIPI_MCR[MOEN]=0, and is exited by writing SIPI_MCR[MOEN]=1 (see SIPI Module Configuration Register (SIPI_MCR)). All the activities on the SIPI Tx and Rx ports are immediately stopped in Module Disable mode.

47.11 Errors

This section describes the potential errors that can occur during SIPI operation.

47.11.1 Timeout error

A timeout error is generated at the initiator node when the acknowledge/response is not received within the time configured in the corresponding CTOR*n*[TOR] field setting (see SIPI Channel Timeout Register 0 (SIPI_CTOR0), SIPI Channel Timeout Register 1 (SIPI_CTOR1), SIPI Channel Timeout Register 2 (SIPI_CTOR2) and SIPI Channel Timeout Register 3 (SIPI_CTOR3)). A timeout error is indicated when ERR[TOE*n*]=1 (see SIPI Error Register (SIPI_ERR)).

Note

SIPI should not drop the response even after a timeout occurs. Software will poll both error and status flags after the transfer to see if there was a timeout error. If there was a timeout error the response received may then be discarded.

47.11.2 CRC error

A CRC error is generated at the target nodes when the CRC received with the frame does not match the calculated CRC, and the SR[GCRCE] is set (see SIPI Status Register (SIPI_SR)). An interrupt will be asserted if MCR[CRCIE]=1 (see SIPI Module Configuration Register (SIPI_MCR)).

Note

The target node will not send an acknowledge to the initiator node when a CRC error is generated. An interrupt will be generated on the target side if the corresponding interrupt enable bit is set. The initiator node will detect a timeout and take necessary action.

47.11.3 Maximum count reached error

The maximum count reached error is only generated at the target node. It is generated when the value of the SIPI_ACR is equal to the SIPI_MAXCR (see SIPI Max Count Register (SIPI_MAXCR) and SIPI Address Count Register (SIPI_ACR)). When the maximum count is reached, SIPI_SR[MCR] = 1 (see SIPI Status Register (SIPI_SR)). An interrupt will be generated if SIPI_MCR[MCRIE] = 1 (see SIPI Module Configuration Register (SIPI_MCR)).

47.11.4 Transaction ID error

The Transaction ID (TID) error is always generated at the initiator node only. It is generated when header bits 15–13 do not match SIPI_CSR*n*[TID] (transaction ID bits, see SIPI Channel Status Register 0 (SIPI_CSR0), SIPI Channel Status Register 1 (SIPI_CSR1), SIPI Channel Status Register 2 (SIPI_CSR2), and SIPI Channel Status Register 3 (SIPI_CSR3)). SIPI_CSR*n*[TIDE] = 1 when a TID error is detected, and an interrupt will be generated if SIPI_CIR*n*[TIDIE] = 1 (see SIPI Channel Interrupt Register 0 (SIPI_CIR0), SIPI Channel Interrupt Register 1 (SIPI_CIR1), SIPI Channel Interrupt Register 2 (SIPI_CIR2), and SIPI Channel Interrupt Register 3 (SIPI_CIR3)).

47.11.5 Acknowledge error

An incorrect acknowledge is received only from the initiator. When the acknowledge received is incorrect, SIPI_ERR[ACKR*n*] will be set (see SIPI Error Register (SIPI_ERR)). An interrupt will be generated if SIPI_CIR*n*[WAIE]=1 (see SIPI Channel Interrupt Register 0 (SIPI_CIR0), SIPI Channel Interrupt Register 1 (SIPI_CIR1), SIPI Channel Interrupt Register 2 (SIPI_CIR2), and SIPI Channel Interrupt Register 3 (SIPI_CIR3)).

47.12 CRC calculation

Example: If header is AABBh, address is 1122_3344h and data is CCDD_EEFFh. Then CRC calculation will take place as follows:

1) The CRC seed is initialized by FFFF_FFFFh.

2) All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).

3) So the header will be sent as DD55_0000h.

4) Address will be sent as 22CC_4488h.

5) Data will be sent as FF77_BB33h.

47.13 Interrupt logic

A description of the interrupt logic can be found in the following figure.

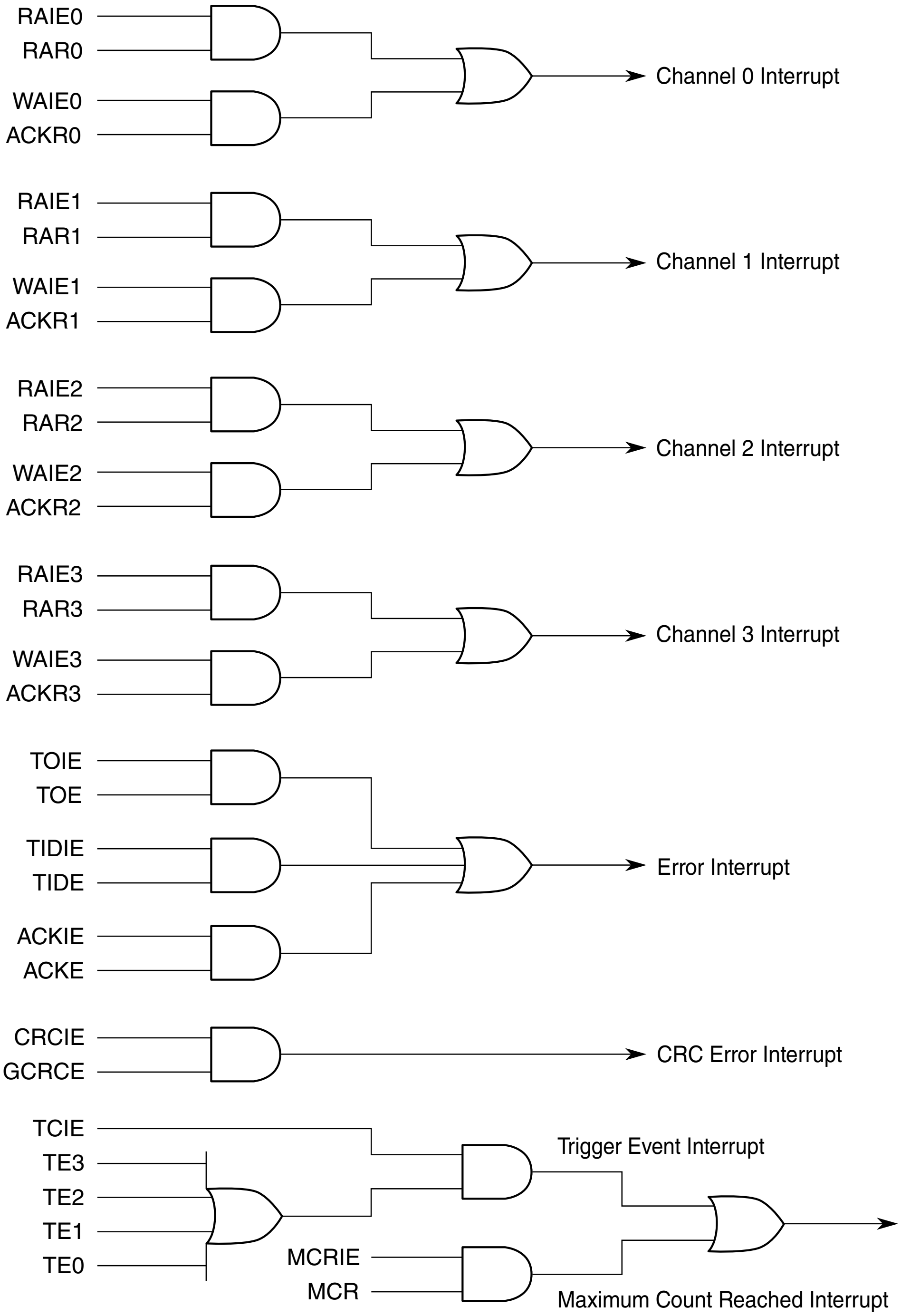


Figure 47-29. Interrupt description

47.14 SIPI control and status overview

The diagram below shows the relationship between transfers and the SIPI control and status registers.

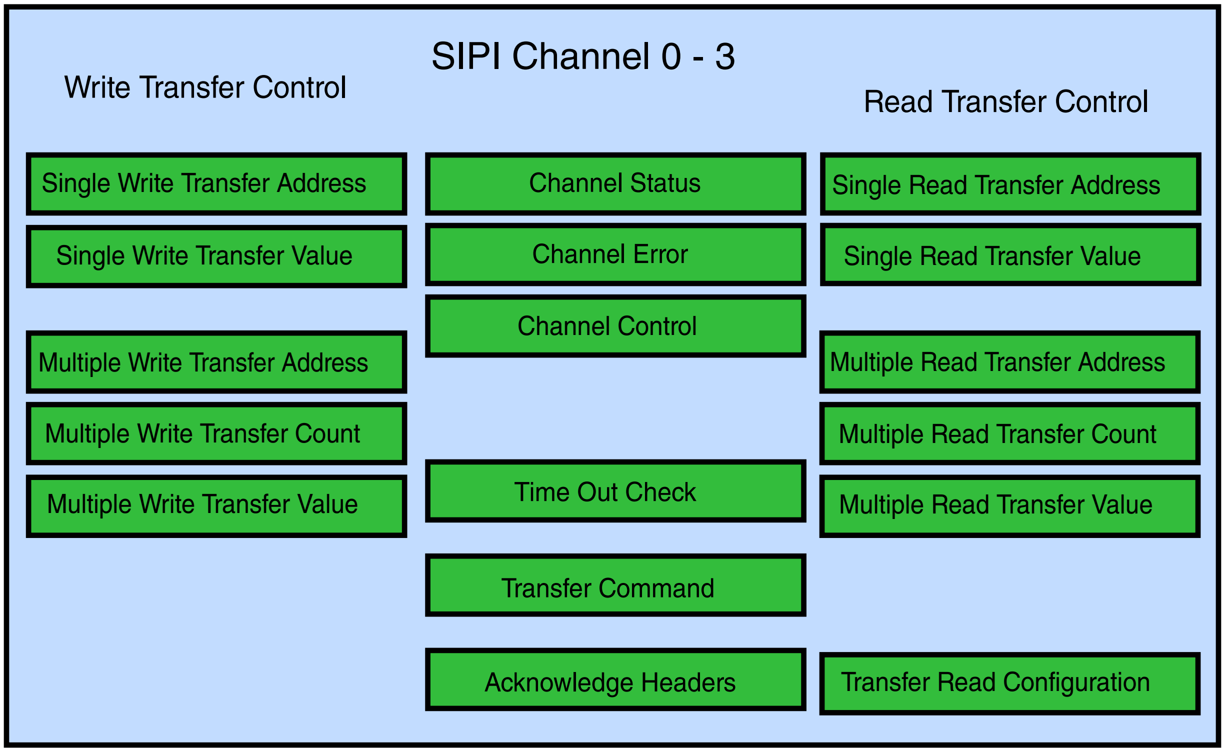

SIPI Channel 0 - 3
Write Transfer Control
Read Transfer Control
Single Write Transfer Address
Single Write Transfer Value
Multiple Write Transfer Address
Multiple Write Transfer Count
Multiple Write Transfer Value
Channel Status
Channel Error
Channel Control
Time Out Check
Transfer Command
Acknowledge Headers
Single Read Transfer Address
Single Read Transfer Value
Multiple Read Transfer Address
Multiple Read Transfer Count
Multiple Read Transfer Value
Transfer Read Configuration

Figure 47-30. SIPI control and status overview – Register transfer

47.15 Memory map and register definition

SIPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SIPI Channel Control Register 0 (SIPI_CCR0)	32	R/W	0000_0000h	47.15.1/ 1443
4	SIPI Channel Status Register 0 (SIPI_CSR0)	32	R	0000_0000h	47.15.2/ 1446
C	SIPI Channel Interrupt Register 0 (SIPI_CIR0)	32	R/W	0000_0000h	47.15.3/ 1447
10	SIPI Channel Timeout Register 0 (SIPI_CTOR0)	32	R/W	0000_00FFh	47.15.4/ 1448
14	SIPI Channel CRC Register 0 (SIPI_CCRC0)	32	R	0000_0000h	47.15.5/ 1449
18	SIPI Channel Address Register 0 (SIPI_CAR0)	32	R/W	0000_0000h	47.15.6/ 1449
1C	SIPI Channel Data Register 0 (SIPI_CDR0)	32	R/W	0000_0000h	47.15.7/ 1450
20	SIPI Channel Control Register 1 (SIPI_CCR1)	32	R/W	0000_0000h	47.15.8/ 1450

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24	SIPI Channel Status Register 1 (SIPI_CSR1)	32	R	0000_0000h	47.15.9/ 1453
2C	SIPI Channel Interrupt Register 1 (SIPI_CIR1)	32	R/W	0000_0000h	47.15.10/ 1455
30	SIPI Channel Timeout Register 1 (SIPI_CTOR1)	32	R/W	0000_00FFh	47.15.11/ 1456
34	SIPI Channel CRC Register 1 (SIPI_CCRC1)	32	R	0000_0000h	47.15.12/ 1457
38	SIPI Channel Address Register 1 (SIPI_CAR1)	32	R/W	0000_0000h	47.15.13/ 1457
3C	SIPI Channel Data Register 1 (SIPI_CDR1)	32	R/W	0000_0000h	47.15.14/ 1458
40	SIPI Channel Control Register 2 (SIPI_CCR2)	32	R/W	0000_0000h	47.15.15/ 1458
44	SIPI Channel Status Register 2 (SIPI_CSR2)	32	R	0000_0000h	47.15.16/ 1461
4C	SIPI Channel Interrupt Register 2 (SIPI_CIR2)	32	R/W	0000_0000h	47.15.17/ 1463
50	SIPI Channel Timeout Register 2 (SIPI_CTOR2)	32	R/W	0000_00FFh	47.15.18/ 1464
54	SIPI Channel CRC Register 2 (SIPI_CCRC2)	32	R	0000_0000h	47.15.19/ 1465
58	SIPI Channel Address Register 2 (SIPI_CAR2)	32	R/W	0000_0000h	47.15.20/ 1465
5C	SIPI Channel Data Register 2 (SIPI_CDR2_0)	32	R/W	0000_0000h	47.15.21/ 1466
60	SIPI Channel Data Register 2 (SIPI_CDR2_1)	32	R/W	0000_0000h	47.15.21/ 1466
64	SIPI Channel Data Register 2 (SIPI_CDR2_2)	32	R/W	0000_0000h	47.15.21/ 1466
68	SIPI Channel Data Register 2 (SIPI_CDR2_3)	32	R/W	0000_0000h	47.15.21/ 1466
6C	SIPI Channel Data Register 2 (SIPI_CDR2_4)	32	R/W	0000_0000h	47.15.21/ 1466
70	SIPI Channel Data Register 2 (SIPI_CDR2_5)	32	R/W	0000_0000h	47.15.21/ 1466
74	SIPI Channel Data Register 2 (SIPI_CDR2_6)	32	R/W	0000_0000h	47.15.21/ 1466
78	SIPI Channel Data Register 2 (SIPI_CDR2_7)	32	R/W	0000_0000h	47.15.21/ 1466
7C	SIPI Channel Control Register 3 (SIPI_CCR3)	32	R/W	0000_0000h	47.15.22/ 1466
80	SIPI Channel Status Register 3 (SIPI_CSR3)	32	R	0000_0000h	47.15.23/ 1470

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	SIPI Channel Interrupt Register 3 (SIPI_CIR3)	32	R/W	0000_0000h	47.15.24/ 1471
8C	SIPI Channel Timeout Register 3 (SIPI_CTOR3)	32	R/W	0000_00FFh	47.15.25/ 1472
90	SIPI Channel CRC Register 3 (SIPI_CCRC3)	32	R	0000_0000h	47.15.26/ 1473
94	SIPI Channel Address Register 3 (SIPI_CAR3)	32	R/W	0000_0000h	47.15.27/ 1473
98	SIPI Channel Data Register 3 (SIPI_CDR3)	32	R/W	0000_0000h	47.15.28/ 1474
9C	SIPI Module Configuration Register (SIPI_MCR)	32	R/W	See section	47.15.29/ 1475
A0	SIPI Status Register (SIPI_SR)	32	R	0000_0000h	47.15.30/ 1478
A4	SIPI Max Count Register (SIPI_MAXCR)	32	R/W	FFFF_FFFCh	47.15.31/ 1480
A8	SIPI Address Reload Register (SIPI_ARR)	32	R/W	0000_0000h	47.15.32/ 1480
AC	SIPI Address Count Register (SIPI_ACR)	32	R/W	0000_0000h	47.15.33/ 1481
B0	SIPI Error Register (SIPI_ERR)	32	R	0000_0000h	47.15.34/ 1482

47.15.1 SIPI Channel Control Register 0 (SIPI_CCR0)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR0[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR0[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 0 (SIPI_CSR0)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 0 (SIPI_CAR0)).

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR0 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR0. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority.

Table continues on the next page...

SIPI_CCR0 field descriptions (continued)

Field	Description
	0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1 or SIPI_CCR0[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

47.15.2 SIPI Channel Status Register 0 (SIPI_CSR0)

CSR0 contains the status bits for the current transfer.

Address: 0h base + 4h offset = 4h

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>R</td><td colspan="16">Reserved</td></tr>
<tr><td>W</td><td colspan="16"></td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

<table>
<tr><td>Bit</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>R</td><td colspan="8">Reserved</td><td>RAR</td><td colspan="3">TID</td><td>ACKR</td><td>CB</td><td colspan="2">Reserved</td></tr>
<tr><td>W</td><td colspan="8"></td><td>w1c</td><td colspan="3">w1c</td><td>w1c</td><td></td><td colspan="2"></td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

SIPI_CSR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 0 status. 0 Channel 0 free 1 Channel 0 busy

Table continues on the next page...

SIPI_CSR0 field descriptions (continued)

Field	Description
30–31 Reserved	This field is reserved.

47.15.3 SIPI Channel Interrupt Register 0 (SIPI_CIR0)

SIPI_CIR0 contains the interrupt enable bits for channel 0.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved										WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CIR0 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

SIPI_CIR0 field descriptions (continued)

Field	Description
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

47.15.4 SIPI Channel Timeout Register 0 (SIPI_CTOR0)

SIPI_CTOR0 contains the timeout value for Tx requests.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

47.15.5 SIPI Channel CRC Register 0 (SIPI_CCRC0)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC0 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

47.15.6 SIPI Channel Address Register 0 (SIPI_CAR0)

SIPI_CAR0 is the address target for data transmission.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR0 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

47.15.7 SIPI Channel Data Register 0 (SIPI_CDR0)

SIPI_CDR0 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR0 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

47.15.8 SIPI Channel Control Register 1 (SIPI_CCR1)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR1[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR1[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 1 (SIPI_CSR1)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 1 (SIPI_CAR1)).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR1 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR1. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR1 field descriptions (continued)

Field	Description
	0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1 or SIPI_CCR1[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

47.15.9 SIPI Channel Status Register 1 (SIPI_CSR1)

CSR1 contains the status bits for the current transfer.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								RAR	TID			ACKR	CB	Reserved	
W									w1c	w1c			w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CSR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 1 status. 0 Channel 1 free 1 Channel 1 busy
30–31 Reserved	This field is reserved.

47.15.10 SIPI Channel Interrupt Register 1 (SIPI_CIR1)

The CIR1 contains the interrupt enable bits for channel 1.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved										WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CIR1 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR1 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

47.15.11 SIPI Channel Timeout Register 1 (SIPI_CTOR1)

SIPI_CTOR1 contains the timeout value for Tx requests.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

47.15.12 SIPI Channel CRC Register 1 (SIPI_CCRC1)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC1 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

47.15.13 SIPI Channel Address Register 1 (SIPI_CAR1)

SIPI_CAR1 is the address target for data transmission.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR1 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

47.15.14 SIPI Channel Data Register 1 (SIPI_CDR1)

SIPI_CDR1 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR1 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

47.15.15 SIPI Channel Control Register 2 (SIPI_CCR2)

NOTE

> Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR2[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR2[CB] = 0).
>
> PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 2 (SIPI_CSR2)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 2 (SIPI_CAR2)).

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR2 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR2. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR2 field descriptions (continued)

Field	Description
	0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1 or SIPI_CCR2[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

47.15.16 SIPI Channel Status Register 2 (SIPI_CSR2)

SIPI_CSR2 contains the status bits for the current transfer.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								RAR	TID			ACKR	CB	Reserved	
W									w1c	w1c			w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CSR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 2 status. 0 Channel 2 free 1 Channel 2 busy
30–31 Reserved	This field is reserved.

47.15.17 SIPI Channel Interrupt Register 2 (SIPI_CIR2)

The SIPI_CIR2 contains the interrupt enable bits for channel 2.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved										WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CIR2 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR2 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

47.15.18 SIPI Channel Timeout Register 2 (SIPI_CTOR2)

SIPI_CTOR2 contains the timeout value for Tx requests.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

47.15.19 SIPI Channel CRC Register 2 (SIPI_CCRC2)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC2 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

47.15.20 SIPI Channel Address Register 2 (SIPI_CAR2)

SIPI_CAR2 is the address target for data transmission. For streaming operations this register is the start address.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR2 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

47.15.21 SIPI Channel Data Register 2 (SIPI_CDR2_*n*)

SIPI_CDR2_*n*contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 5Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR2_*n* field descriptions

Field	Description
0–31 CDR2	Data register bits. Contains the data that will be transmitted, or received.

47.15.22 SIPI Channel Control Register 3 (SIPI_CCR3)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR3[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR3[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 3 (SIPI_CSR3)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 3 (SIPI_CAR3)).

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR3 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR3. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT

Table continues on the next page...

SIPI_CCR3 field descriptions (continued)

Field	Description
	3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1 or SIPI_CCR3[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

47.15.23 SIPI Channel Status Register 3 (SIPI_CSR3)

SIPI_CSR3 contains the status bits for the current transfer.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								RAR	TID			ACKR	CB	Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CSR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received

Table continues on the next page...

SIPI_CSR3 field descriptions (continued)

Field	Description
29 CB	Channel Busy. Indicates channel 3 status. 0 Channel 3 free 1 Channel 3 busy
30–31 Reserved	This field is reserved.

47.15.24 SIPI Channel Interrupt Register 3 (SIPI_CIR3)

SIPI_CIR3 contains the interrupt enable bits for channel 3.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved										WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CIR3 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable.

Table continues on the next page...

SIPI_CIR3 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

47.15.25 SIPI Channel Timeout Register 3 (SIPI_CTOR3)

SIPI_CTOR3 contains the timeout value for Tx requests.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

47.15.26 SIPI Channel CRC Register 3 (SIPI_CCRC3)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC3 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

47.15.27 SIPI Channel Address Register 3 (SIPI_CAR3)

SIPI_CAR3 is the address target for data transmission.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR3 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

47.15.28 SIPI Channel Data Register 3 (SIPI_CDR3)

SIPI_CDR3 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR3 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

47.15.29 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a global 32-bit configuration register.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZ	Reserved	HALT	Reserved		PRSCLR										
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AID		Reserved			CRCIE	MCRIE	Reserved				CHNSB	TEN	INIT	MOEN	SR
W																
Reset	0*	0*	0	0	0	0*	0*	0	0	0	0	0*	0*	0*	0	0

* Notes:
- INIT field: Can only be written when SIPI_MCR[MOEN] = 1.
- TEN field: Can only be written when SIPI_MCR[MOEN] = 1.
- CHNSB field: Can be written only once after reset.
- MCRIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- CRCIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- AID field: Can be written in initialization mode only (SIPI_MCR[INIT] = 1).

SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1.

Table continues on the next page...

SIPI_MCR field descriptions (continued)

Field	Description
	0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
1 Reserved	This field is reserved. **Important:** Always write the default value to this field.
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1. 0 No Freeze mode request 1 Enters Freeze mode if FRZ bit is asserted.
3–4 Reserved	This field is reserved.
5–15 PRSCLR	Timeout counter prescaler The timeout counter runs on the prescaled system clock. Default value is 64 (040h). The allowed programmable values are (all other values are ignored): 040h 64 (default) 080h 128 100h 256 200h 512 400h 1024 **NOTE:** This field should be programmed by software during initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. **NOTE:** Writes to SIPI_MCR[PRSCLR] can only be accomplished with 16-bit or 32-bit writes.
16–17 AID	Address Increment/Decrement These bits define the type of address change at the target node. **NOTE:** This bits should be programmed by software in initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. 00 no change. address stays same 01 address increments by 4 10 address decrements by 4 11 not used
18–20 Reserved	This field is reserved.
21 CRCIE	CRC Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
22 MCRIE	Max Count Reached Interrupt Enable

Table continues on the next page...

SIPI_MCR field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
23–26 Reserved	This field is reserved.
27 CHNSB	Channel coding select bit. 0 Code Table II (see Table 47-2) 1 Code Table I (see Table 47-2)
28 TEN	Target Enable Setting this bit enables the target functionality at SIPI. This bit can be read or written anytime. This bit is automatically negated by hardware when target detects an error in "streaming without ACK" mode. This bit has to be enabled for the transmission operations also.
29 INIT	Initialization Mode Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits can't be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, all SIPI operations are immediately stopped.
31 SR	Soft Reset Setting this bit clears all status and error registers, and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete.

47.15.30 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZACK	LPMACK	Reserved													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					GCRCE	MCR	Reserved	TE				STATE			
W						w1c	w1c		w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge. This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited.

Table continues on the next page...

SIPI_SR field descriptions (continued)

Field	Description
	0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge. This read-only bit indicates that SIPI is in Disable Mode. Disable mode can not be entered until all current transmission or reception processes have finished. The CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI is not in low power mode 1 SIPI is in Disable Mode.
2–20 Reserved	This field is reserved.
21 GCRCE	Global CRC Error Bit. 0 No CRC error 1 CRC error occurred
22 MCR	Maximum Count Reached. This bit will be set whenever SIPI_ACR[ADCNT] = SIPI_MAXCR[MXCNT]. An interrupt will be generated when this bit is set only if SIPI_MCR[MCRIE] = 1. This it should be cleared by software.
23 Reserved	This field is reserved.
24–27 TE	Trigger Event on Respective Channels. This field enables interrupts for target nodes. xxx1 TE0 = 1 - Channel 0 trigger event xx1x TE1 = 1 - Channel 1 trigger event x1xx TE2 = 1 - Channel 2 trigger event 1xxx TE3 = 1 - Channel 3 trigger event
28–31 STATE	These bits reflect the transmit state machine status. They can be polled for determination of state machine status. 0000 IDLE 0001 HEADER_AND_ADDRESS_FIELD 0010 HEADER_AND_DATA_FIELD 0011 HEADER_AND_CRC_FIELD 0100 ADDRESS_AND_CRC_FIELD 0101 ADDRESS_AND_DATA_FIELD 0110 DATA_AND_CRC_FIELD 0111 DATA_FIELD

47.15.31 SIPI Max Count Register (SIPI_MAXCR)

SIPI_MAXCR contains the maximum address count value at target node. It is programmed via direct write request through the initiator. This register can be read or written by software anytime.

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MXCNT															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MXCNT														Reserved	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SIPI_MAXCR field descriptions

Field	Description
0–29 MXCNT	This field contains the maximum address count value at the target node. It should be programmed via direct write request through the initiator.
30–31 Reserved	This field is reserved.

47.15.32 SIPI Address Reload Register (SIPI_ARR)

The SIPI_ARR contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.

NOTE

This register is writeable only when SIPI_MCR[INIT] = 1.

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADRLD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADRLD														Reserved	
W															▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ARR field descriptions

Field	Description
0–29 ADRLD	ADRLD contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.
30–31 Reserved	This field is reserved.

47.15.33 SIPI Address Count Register (SIPI_ACR)

This register reflects the count value of address counter at target node. It should be configured by direct write request from initiator. This register can be read/written by software anytime.

NOTE

SIPI_ARR, SIPI_ACR and SIPI_MAXCR will be configured by direct write operation through the initiator.

NOTE

When a streaming write command is received, the target will start the write operation from the address present in SIPI_ACR (address count register).

NOTE

After each 32-bit write has completed, SIPI_ACR[ADCNT] is compared against SIPI_MAXCR[MXCNT]. If they are equal, SIPI_ACR[ADCNT] is loaded with the value stored in SIPI_ARR[ADRLD]. If they are not equal, the SIPI_ACR[ADCNT] will increment by 4, decrement by 4 or remain the same (depending on configuration). In both cases, the SIPI_ACR[ADCNT] will contain the address at which the next streaming write will occur.

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADCNT														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ACR field descriptions

Field	Description
0–29 ADCNT	Feflects the count value of address counter at target node. It should be configured by direct write request from the initiator. This register can be read/written by software anytime. At the end of the streaming operation, SIPI_ACR will point to the next address to be written.
30–31 Reserved	This field is reserved.

47.15.34 SIPI Error Register (SIPI_ERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate errors.

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					TOE3	TIDE3	ACKE3	Reserved					TOE2	TIDE2	ACKE2
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					TOE1	TIDE1	ACKE1	Reserved					TOE0	TIDE0	ACKE0
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ERR field descriptions

Field	Description
0–4 Reserved	This field is reserved.
5 TOE3	Timeout Error for Channel 3. 0 Timeout error occured 1 Timeout error didn't occur
6 TIDE3	Transaction ID Error for Channel 3. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
7 ACKE3	Acknowledge Error for Channel 3. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
8–12 Reserved	This field is reserved.
13 TOE2	Timeout Error for Channel 2. 0 Timeout error occured 1 Timeout error didn't occur
14 TIDE2	Transaction ID Error for Channel 2. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
15 ACKE2	Acknowledge Error for Channel 2. 0 Acknowledge received is correct 1 Acknowledge received is not correct
16–20 Reserved	This field is reserved.
21 TOE1	Timeout Error for Channel 1. 0 Timeout error occured 1 Timeout error didn't occur
22 TIDE1	Transaction ID Error for Channel 1. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
23 ACKE1	Acknowledge Error for Channel 1. 0 Acknowledge received is correct 1 Acknowledge received is not correct
24–28 Reserved	This field is reserved.
29 TOE0	Timeout Error for Channel 0. 0 Timeout error occured 1 Timeout error did not occur

Table continues on the next page...

SIPI_ERR field descriptions (continued)

Field	Description
30 TIDE0	Transaction ID Error for Channel 0. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
31 ACKE0	Acknowledge Error for Channel 0. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

Chapter 48
LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

48.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

This chapter describes the specifications of the LFAST module, which implements the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in dual mode (software configurable master/slave operation) for interprocessor communications.

48.2 Block diagram

The following figure depicts LFAST interaction with other modules on the device.

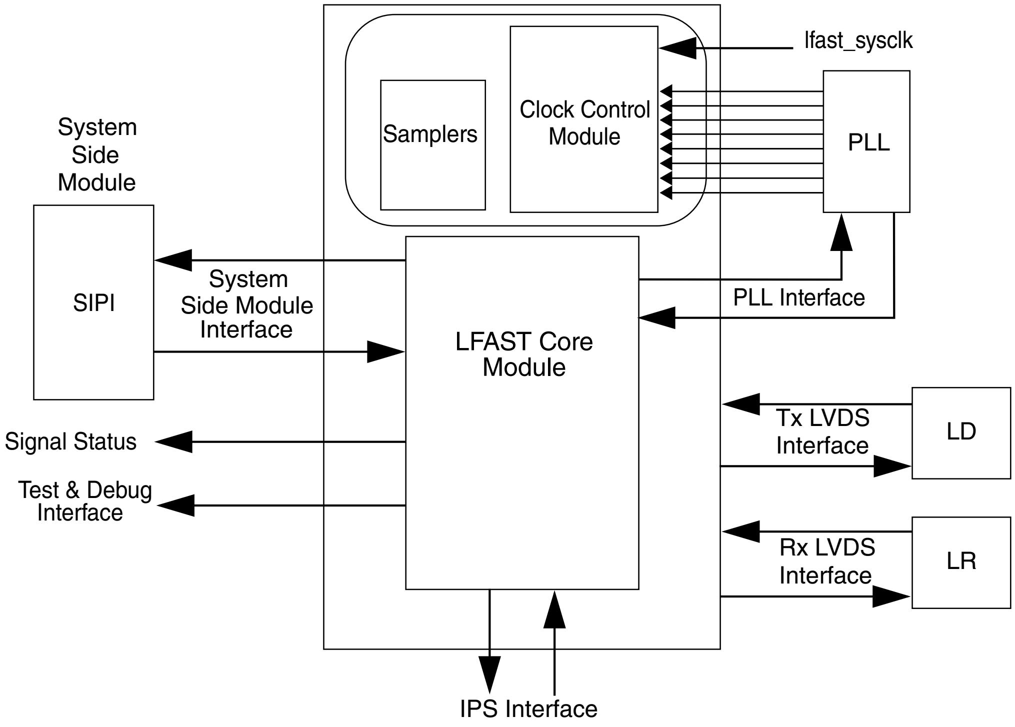


Figure 48-1. LFAST block diagram

48.3 External signals

LFAST is a five pin interface with the following signals :

- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

48.3.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low data rate, and the high data rate is in the Data Sheet.

48.4 LFAST frame structure

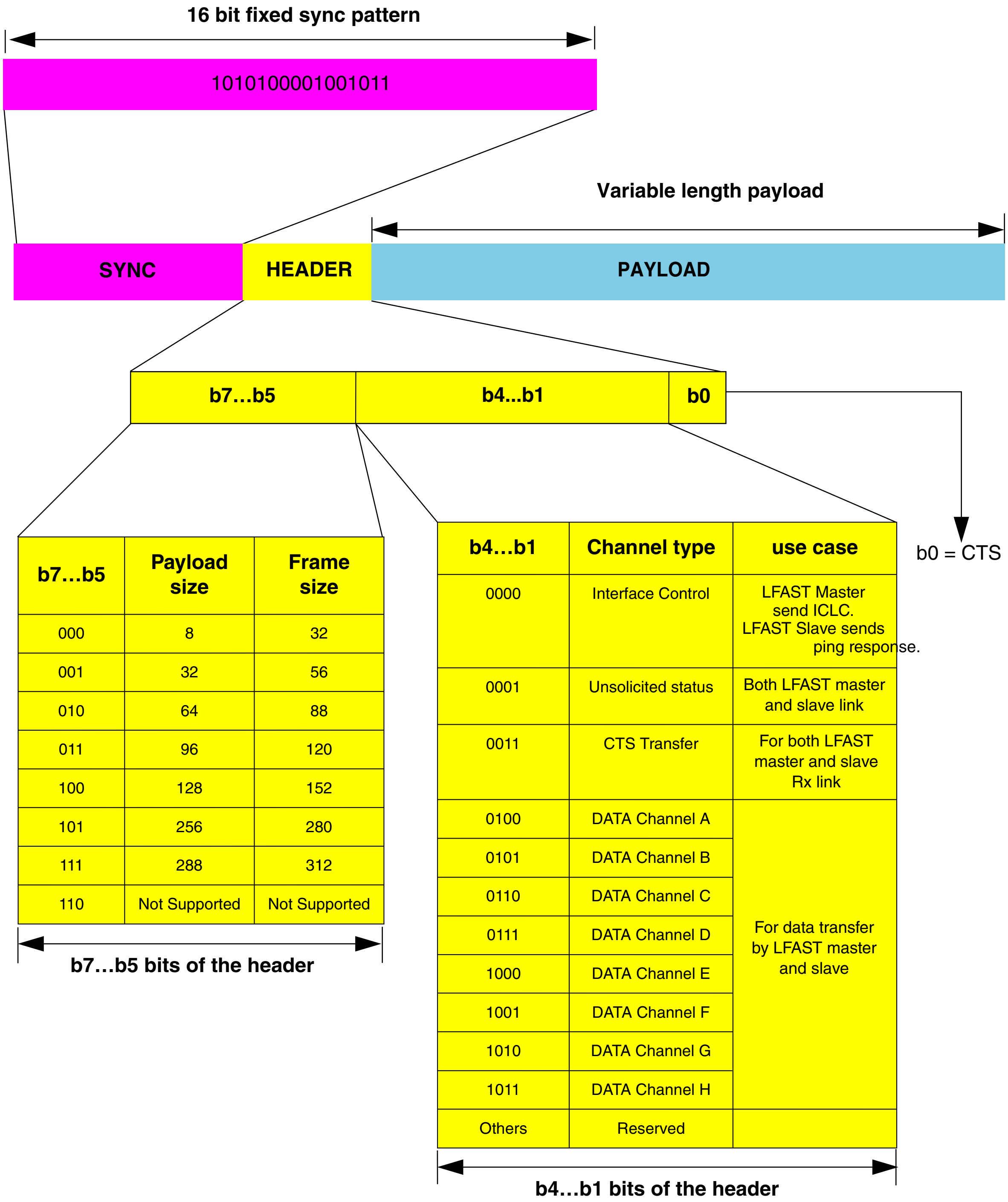
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length.

Sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See Figure 48-2 for details of the LFAST frame.



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	288	312
110	Not Supported	Not Supported

b4…b1	Channel type	use case
0000	Interface Control	LFAST Master send ICLC. LFAST Slave sends ping response.
0001	Unsolicited status	Both LFAST master and slave link
0011	CTS Transfer	For both LFAST master and slave Rx link
0100	DATA Channel A	For data transfer by LFAST master and slave
0101	DATA Channel B	
0110	DATA Channel C	
0111	DATA Channel D	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

Figure 48-2. LFAST frame structure

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

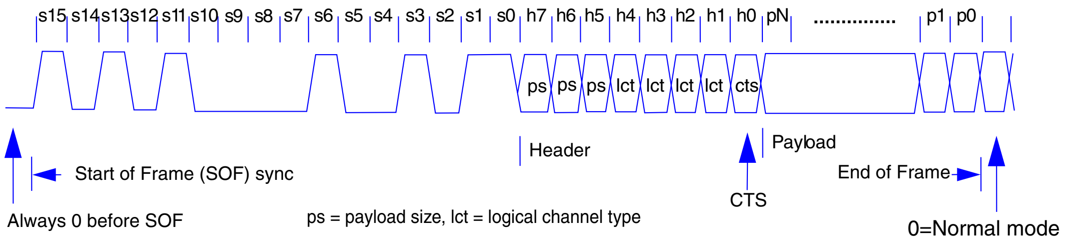

s15 s14 s13 s12 s11 s10 s9 s8 s7 s6 s5 s4 s3 s2 s1 s0 h7 h6 h5 h4 h3 h2 h1 h0 pN p1 p0
ps ps ps lct lct lct lct cts
Header
Payload
CTS
Start of Frame (SOF) sync
Always 0 before SOF
ps = payload size, lct = logical channel type
End of Frame
0=Normal mode

Figure 48-3. Serial frame structure

The frame structure is shown in Figure 48-3 and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size as shown in Table 48-1 :

Table 48-1. Header payload sizes

b7-b5(bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	—	—
111	288	312

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type and clear to send (CTS) frame.
- **Header - (b0):** CTS on the both LFAST master and slave devices.
- **Payload:** Content dependent upon frame type.
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode)

48.5 Features

- Supports dual mode (register configurable Master/Slave).
- Supports asynchronous data transfer up to the maximum data rate shown in the product Data Sheet.
- Transmits and receives data, CTS, ICLC and unsolicited frames.
- Receives ICLC frames
- Provision of five interrupts for Tx and Rx channels.
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports LFAST defined CTS controlled data transfer. No dependency between data transmission and reception unless CTS mode is enabled.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for data frame with variable payload sizes of 32, 64, 96, 128, 256 or 288 bits.
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation in slave mode.
- Supports for detection of unsupported channel number and unsupported payload size.

48.6 Memory map and register definition

All registers are 32 bits wide.

NOTE

Read/Write accesses to all unimplemented registers and write accesses to Read only register will return a Transfer Error.

LFAST memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LFAST Mode Configuration Register (LFAST_MCR)	32	R/W	See section	48.6.1/1493
4	LFAST Speed Control Register (LFAST_SCR)	32	R/W	0001_0000h	48.6.2/1495
8	LFAST Correlator Control Register (LFAST_COCR)	32	R/W	0000_000Eh	48.6.3/1496
C	LFAST Test Mode Control Register (LFAST_TMCR)	32	R/W	0000_0000h	48.6.4/1498
10	LFAST Auto Loopback Control Register (LFAST_ALCR)	32	R/W	0000_0000h	48.6.5/1499
14	LFAST Rate Change Delay Control Register (LFAST_RCDCR)	32	R/W	000F_0000h	48.6.6/1500
18	LFAST Wakeup Delay Control Register (LFAST_SLCR)	32	R/W	1201_5F02h	48.6.7/1500
1C	LFAST ICLC Control Register (LFAST_ICR)	32	R/W	0000_0000h	48.6.8/1502
20	LFAST Ping Control Register (LFAST_PICR)	32	R/W	0000_80CAh	48.6.9/1503
2C	LFAST Rx FIFO CTS Control Register (LFAST_RFCR)	32	R/W	000F_0009h	48.6.10/ 1503
30	LFAST Tx Interrupt Enable Register (LFAST_TIER)	32	R/W	0000_0000h	48.6.11/ 1504
34	LFAST Rx Interrupt Enable Register (LFAST_RIER)	32	R/W	0000_0000h	48.6.12/ 1505
38	LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)	32	R/W	0000_0000h	48.6.13/ 1507
3C	LFAST PLL Control Register (LFAST_PLLCR)	32	R/W	0000_005Ch	48.6.14/ 1509
40	LFAST LVDS Control Register (LFAST_LCR)	32	R/W	See section	48.6.15/ 1511
44	LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)	32	R/W	0000_0000h	48.6.16/ 1514
48	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR0)	32	R/W	0000_0000h	48.6.17/ 1514
4C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR1)	32	R/W	0000_0000h	48.6.17/ 1514
50	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR2)	32	R/W	0000_0000h	48.6.17/ 1514
54	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR3)	32	R/W	0000_0000h	48.6.17/ 1514

Table continues on the next page...

LFAST memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
58	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR4)	32	R/W	0000_0000h	48.6.17/ 1514
5C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR5)	32	R/W	0000_0000h	48.6.17/ 1514
60	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR6)	32	R/W	0000_0000h	48.6.17/ 1514
64	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR7)	32	R/W	0000_0000h	48.6.17/ 1514
68	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR8)	32	R/W	0000_0000h	48.6.17/ 1514
80	LFAST Global Status Register (LFAST_GSR)	32	R	See section	48.6.18/ 1515
84	LFAST Ping Status Register (LFAST_PISR)	32	R	0000_0000h	48.6.19/ 1516
94	LFAST Data Frame Status Register (LFAST_DFSR)	32	R	0000_0000h	48.6.20/ 1517
98	LFAST Tx Interrupt Status Register (LFAST_TISR)	32	R/W	0000_0000h	48.6.21/ 1518
9C	LFAST Rx Interrupt Status Register (LFAST_RISR)	32	R/W	0000_0000h	48.6.22/ 1519
A0	LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)	32	w1c	0000_0000h	48.6.23/ 1521
A4	LFAST PLL and LVDS Status Register (LFAST_PLLLSR)	32	R	0002_0003h	48.6.24/ 1523
A8	LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)	32	R/W	0000_0000h	48.6.25/ 1524
AC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR0)	32	R	0000_0000h	48.6.26/ 1525
B0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR1)	32	R	0000_0000h	48.6.26/ 1525
B4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR2)	32	R	0000_0000h	48.6.26/ 1525
B8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR3)	32	R	0000_0000h	48.6.26/ 1525
BC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR4)	32	R	0000_0000h	48.6.26/ 1525
C0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR5)	32	R	0000_0000h	48.6.26/ 1525
C4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR6)	32	R	0000_0000h	48.6.26/ 1525
C8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR7)	32	R	0000_0000h	48.6.26/ 1525
CC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR8)	32	R	0000_0000h	48.6.26/ 1525

48.6.1 LFAST Mode Configuration Register (LFAST_MCR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSEN	0						IPGDBG	0							LSSEL
W																
Reset	0*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0								TXARBD	CTSEN	0	DRFRST	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0*	0	0*	0

* Notes:

- DRFRST field: Set by user software and then cleared by system hardware.
- CTSEN field: Only writable when MCR[DRFEN] = 0.
- MSEN field: Writable only once after the asynchronous reset.

LFAST_MCR field descriptions

Field	Description
0 MSEN	LFAST Master or Slave mode Enable. This bit selects either the LFAST master or slave functionality. 0 Enable the modules LFAST Slave functionality only. 1 Enable the modules LFAST Master functionality only.
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode (see Slow speed clock for details). 0 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable

Table continues on the next page...

LFAST_MCR field descriptions (continued)

Field	Description
	registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS Line Receiver (LR). 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface isn't disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be modified by LFAST slave H/W on reception of an ICLC command frame. This field can only be written in dual mode (LFAST_GSR[DUALMD] = 1). 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 LFAST transmitter Interface is enabled. New requests are accepted.
19–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 CTSEN	CTS Enable. This bit defines the Push-Pull mode of the LFAST devices receiver. This bit is used to enable/disable CTS mode of the Tx block. This bit is only writable when MCR[DRFEN] = 0. 0 CTS mode is disabled. Indicates that the device is in Push mode. The CTS bit of frames transmitted is 1. The CTS bit doesn't represent the status of Rx FIFO. 1 CTS mode is enabled. The CTS bit of all transmit frames is set when the Rx FIFO empty space is on or above higher threshold, and cleared when the Rx FIFO empty space is on or below lower threshold.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after Reset.. 0 No Soft Reset 1 Soft Reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

48.6.2 LFAST Speed Control Register (LFAST_SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															DRMD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							RDR	0							TDR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_SCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. 0 S/W controls the Data Rate controller mode. In LFAST Slave the ICLC frames for rate change have no affect on the Data rate. 1 In LFAST Slave the reception of ICLC frame for rate change sets appropriate speed mode. In LFAST Master the SCR[DRMD] should be 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. For LFAST Master: • S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Tx interface. For LFAST Slave: • When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Rx interface. • The S/W can program this bit when SCR[DRMD] = 0. • This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is high speed.

Table continues on the next page...

LFAST_SCR field descriptions (continued)

Field	Description
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. For LFAST Master: • S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Rx interface. For LFAST Slave: • When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Tx interface. • The S/W can program this bit when SCR[DRMD] = 0. • This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is high speed.

48.6.3 LFAST Correlator Control Register (LFAST_COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMPSEL								0						0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CORRTH			PHSSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

LFAST_COCR field descriptions

Field	Description
0–7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. This field can only be written when MCR[RXEN] = 0.

Table continues on the next page...

LFAST_COCR field descriptions (continued)

Field	Description
	00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 CORRTH	Correlator threshold level. Defines the correlation threshold level. This field can only be written when MCR[RXEN] = 0. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. This field can only be written when MCR[RXEN] = 0 0 8 phases 1 4 phases

48.6.4 LFAST Test Mode Control Register (LFAST_TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes.

Address: 0h base + Ch offset = Ch

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>R</td><td colspan="6">0</td><td rowspan="2">CLKTST</td><td rowspan="2">LPON</td><td colspan="5">0</td><td colspan="3" rowspan="2">LPMOD</td></tr>
<tr><td>W</td><td colspan="6"></td><td colspan="5"></td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

<table>
<tr><td>Bit</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>R</td><td colspan="16" rowspan="2">LPFRMTH</td></tr>
<tr><td>W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

LFAST_TMCR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1 Loopback mode is enabled 0 Loopback mode is disabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. This field can only be written when TMCR[LPON] = 0. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation 101 Reserved

Table continues on the next page...

LFAST_TMCR field descriptions (continued)

Field	Description
	110 Reserved 111 Reserved
16–31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

48.6.5 LFAST Auto Loopback Control Register (LFAST_ALCR)

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															LPCNTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_ALCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto pre-defined loopback frame transmission. This field can only be written when TMCR[LPON] = 0. 0 Infinite pre-defined loopback frame transmission enabled 1 Fixed count of pre-defined loopback frame transmission enabled
16–31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of pre-defined auto loopback frames to be sent. The pre-defined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload

Table continues on the next page...

LFAST_ALCR field descriptions (continued)

Field	Description
	... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

48.6.6 LFAST Rate Change Delay Control Register (LFAST_RCDCR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DRCNT				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RCDCR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. This field can only be written when MCR[DRFEN] = 1. Number of cycles = DRCNT value.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.6.7 LFAST Wakeup Delay Control Register (LFAST_SLCR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HSCNT								0				LSCNT				HWKCNT								0				LWKCNT			
W																																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

LFAST_SLCR field descriptions

Field	Description
0–7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycle 01h 1 cycle ...

Table continues on the next page...

LFAST_SLCR field descriptions (continued)

Field	Description
	12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16–23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycles 01h 1 cycle ... 5Fh 95 cycles (1.18 µs) ... FFh 255 cycles Maximum
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle 2h 2 cycles (1.18 µs) ... Eh 14 cycles Fh 15 cycles

48.6.8 LFAST ICLC Control Register (LFAST_ICR)

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														ICLCSEQ	SNDICLC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								ICLCPLD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_ICR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 ICLCSEQ	ICLC enabled. This bit should be set whenever the S/W is performing a series of ICLC frame transfers. This bit ensures only ICLC frame transmission request is serviced, other pending frames request are ignored. 0 Single ICLC frame transfer. 1 S/W is performing ICLC frame transfers. Only the ICLC frames will be scheduled during this period. All the other frames will be scheduled only after ICR[ICLCSEQ] = 0.
15 SNDICLC	ICLC frame request. This bit is set to initiate the transfer of ICLC frame by LFAST master. This bit should be set (ICR[SNDICLC] = 1) after writing the required ICLC payload to be transmitted in the ICLCPLD field of the register. This bit is self clearing, which will be cleared when ICLC frame transfer is complete. Set by user software and cleared by system hardware. 0 No Valid ICLC frame for transfer. 1 Valid ICLC frame for transfer.
16–23 Reserved	This field is reserved.
24–31 ICLCPLD	ICLC Payload. This field is used to program the payload of the ICLC frame to be transmitted. New ICLC payload should be set when ICR[SNDICLC] = 0. This field can only be written when ICR[SNDICLC] = 0 (see Table 48-8 for supported ICLC payloads).

48.6.9 LFAST Ping Control Register (LFAST_PICR)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															PNGREQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PNGAUTO	0							PNGPYLD							
W																
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

LFAST_PICR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. Set by user software and cleared by system hardware. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master. This field can only be written when MCR[DRFEN] = 0. 0 Ping response should not be automatically sent. 1 Ping response should be automatically sent.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PNGPYLD	LFAST Slave: Defines the LFAST slaves ping reply frame payload content. This field can only be written when MCR[DRFEN] = 0. LFAST Master: Defines the expected payload of ping response frame. Used for comparison with ping frame received.

48.6.10 LFAST Rx FIFO CTS Control Register (LFAST_RFCR)

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										RCTSMX						0										RCTSMN					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

LFAST_RFCR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 RCTSMX	Rx FIFO Maximum Threshold. Defines the condition for CTS bit of frames to be negated. This field can only be written when LFAST_MCR[DRFEN] = 0.
16–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 RCTSMN	Rx FIFO Minimum Threshold. Defines the condition for CTS bit of frames to be set. This field can only be written when LFAST_MCR[DRFEN] = 0.

48.6.11 LFAST Tx Interrupt Enable Register (LFAST_TIER)

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														TXIIE	TXOVIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											TXPNGIE	0	TXUNSIE	TXICLCIE	TXDTIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_TIER field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIIE	Tx Data Interface Not Enabled - (Mask) Enables or disables the interrupt. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt is disabled 1 Interrupt is enabled
15 TXOVIE	Transmit Data FIFO Overflow Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGIE	Ping Response Frame Transmitted Interrupt Enable.

Table continues on the next page...

LFAST_TIER field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TXUNSIE	Unsolicited Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
30 TXICLCIE	ICLC Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
31 TXDTIE	Data Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled

48.6.12 LFAST Rx Interrupt Enable Register (LFAST_RIER)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOIE	RXMNIE	RXMXIE	RXUFIE	RXOFIE	RXSZIE	RXICIE	RXLCEIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSIE	RXDIE	RXUNSIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIER field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RXUOIE	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

LFAST_RIER field descriptions (continued)

Field	Description
9 RXMNIE	Rx Data FIFO Min Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
10 RXMXIE	Rx Data FIFO Max Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
11 RXUFIE	Rx Data FIFO Underflow 0 Interrupt is disabled 1 Interrupt is enabled
12 RXOFIE	Rx Data FIFO Overflow 0 Interrupt is disabled 1 Interrupt is enabled
13 RXSZIE	Frame with unsupported frame size received. Valid frame sizes are defined in Table 48-12 0 Interrupt is disabled 1 Interrupt is enabled
14 RXICIE	Invalid ICLC code Received 0 Interrupt is disabled 1 Interrupt is enabled
15 RXLCEIE	Invalid Logical Channel Type 0 Interrupt is disabled 1 Interrupt is enabled
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSIE	Frame with CTS bit Low Received 0 Interrupt is disabled 1 Interrupt is enabled
29 RXDIE	Data frame received 0 Interrupt is disabled 1 Interrupt is enabled
30 RXUNSIE	Unsolicited Frame received 0 Interrupt is disabled 1 Interrupt is enabled
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.6.13 LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFIE	ICPSIE	ICPRIE	ICTOIE	ICLPIE	ICCTIE	ICTDIE	ICTEIE	ICRFIE	ICRSIE	ICTFIE	ICTSIE	ICPOFIE	ICPONIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIIER field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFIE	Ping Frame Response failed 0 Interrupt is disabled 1 Interrupt is enabled
19 ICPSIE	Ping Frame Response successful 0 Interrupt is disabled 1 Interrupt is enabled
20 ICPRIE	ICLC frame for Ping Frame Request received 0 Interrupt is disabled 1 Interrupt is enabled
21 ICTOIE	ICLC frame for Test mode off received 0 Interrupt is disabled 1 Interrupt is enabled
22 ICLPIE	ICLC frame for Loopback On received 0 Interrupt is disabled 1 Interrupt is enabled
23 ICCTIE	ICLC frame for Clk Test mode on received 0 Interrupt is disabled 1 Interrupt is enabled
24 ICTDIE	ICLC frame for LFAST Slaves Tx Interface Disable received

Table continues on the next page...

LFAST_RIIER field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
25 ICTEIE	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt is disabled 1 Interrupt is enabled
26 ICRFIE	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
27 ICRSIE	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
28 ICTFIE	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
29 ICTSIE	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
30 ICPOFIE	ICLC frame for PLL OFF received 0 Interrupt is disabled 1 Interrupt is enabled
31 ICPONIE	ICLC frame for PLL ON received 0 Interrupt is disabled 1 Interrupt is enabled

48.6.14 LFAST PLL Control Register (LFAST_PLLCR)

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPTMOD			0											SWPOFF	SWPON
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REFINV	LPCFG		0		PLCKCW		FDIVEN	FBDIV						PREDIV	
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0

LFAST_PLLCR field descriptions

Field	Description
0–2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 1 010 Force Vctrl 011 Charge Pump Up 100 Charge Pump Up Internal Test 101 Charge Pump Idle 110 Charge Pump Down 111 Closed Loop 2
3–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SWPOFF	SW signal to turn OFF the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned OFF.
15 SWPON	SW signal to turn ON the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned ON
16 REFINV	Inverts reference clock edge to PFD. If System PLL PFD using same reference clock, enabling this feature will minimize noise injection crosstalk via the substrate. 0 Do not inverted 1 Invert

Table continues on the next page...

LFAST_PLLCR field descriptions (continued)

Field	Description
17–18 LPCFG	PLL Loop Optimization for sysclk frequency 00 1 × IBASE current 01 0.5 × IBASE current 10 1.5 × IBASE current 11 2 × IBASE current
19–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 PLCKCW	PLL Lock Ready Count Width. Defines the number of cycles PLL waits for before asserting lock_ready flag High 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider. Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24–29 FBDIV	Feedback Division factor for VCO output clock. This field can only be written when LFAST_MCR[DRFEN] = 0. 00h No clock output 01h – 0Ah Reserved 0Bh Divide by 12 (11.5, if FDIVEN = 1) ... ... 1Fh Divide by 32 (31.5, if FDIVEN = 1) 20h – 3Fh Reserved
30–31 PREDIV	Division factor for PLL Reference Clock input. This field can only be written when LFAST_MCR[DRFEN] = 0. 00 Direct clock passed 01 Divide by 2 10 Divide by 3 11 Divide by 4

48.6.15 LFAST LVDS Control Register (LFAST_LCR)

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								SWWKLD	SWSLPLD	SWWKLR	SWSLPLR	SWOFFLD	SWONLD	SWOFFLR	SWONLR
W																
Reset	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0					LVRXOP_TR	Reserved	LVRXOP_BR	LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

* Notes:

- SWONLR field: Set by user software and cleared by system hardware.
- SWOFFLR field: Set by user software and cleared by system hardware.
- SWONLD field: Set by user software and cleared by system hardware.
- SWOFFLD field: Set by user software and cleared by system hardware.
- SWSLPLR field: Set by user software and cleared by system hardware.
- SWWKLR field: Set by user software and cleared by system hardware.
- SWSLPLD field: Set by user software and cleared by system hardware.
- SWWKLD field: Set by user software and cleared by system hardware.

LFAST_LCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be put in sleep

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned OFF(provided no other source is trying to turn the LD ON)
13 SWONLD	SW signal to turn ON the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned ON
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned OFF (provided no other source is trying to turn the LR ON)
15 SWONLR	SW signal to turn ON the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned ON
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
20 LVLPEN	Tx LVDS internal loopback enable The bit is reflected by output signal ipp_digrf_lvds_lpbk_en.The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only): 0 Disable on-chip LFAST receiver termination 1 Enable on-chip LFAST receiver termination
27 Reserved	This field is reserved.
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only): 0 Use for LFAST receiver baud rates less than the maximum baud rate. 1 Required for LFAST receiver maximum baud rate.
29 LVTXOP	Control signal for LFAST selection. **NOTE:** This field must be written 1 for correct operation. 0 Reserved 1 LFAST Bus selection
30 LVCKSS	LVDS Clock Sync Select This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 normal clock used to sample the LVDS data 1 adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

48.6.16 LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															USNDRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									UNSHDR						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSTCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 USNDRQ	Tx Unsolicited send request. Set by user software and cleared by system hardware. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 UNSHDR	Tx Unsolicited message header. This field can only be written when LFAST_UNSTCR[USNDRQ] = 0

48.6.17 LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR*n*)

Address: 0h base + 48h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	UNTXD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSTDR*n* field descriptions

Field	Description
0–31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

48.6.18 LFAST Global Status Register (LFAST_GSR)

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0												LRMD	LDSM	DRSM
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											LPTXDN	LPFPDV	LPCPDV	LPCHDV	LPCSDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module in Slave only mode 1 LFAST Module in Dual mode
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active.

Table continues on the next page...

LFAST_GSR field descriptions (continued)

Field	Description
	0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by LFAST_ALCR[LPFMCNT]. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

48.6.19 LFAST Ping Status Register (LFAST_PISR)

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								RXPNGD							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_PISR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RXPNGD	Ping Data Register. In LFAST Master mode the Ping response ICLC frame received is stored into this register.

48.6.20 LFAST Data Frame Status Register (LFAST_DFSR)

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		RXDCNT						0					RXFCNT			0		TXDCNT						0					TXFCNT		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_DFSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

48.6.21 LFAST Tx Interrupt Status Register (LFAST_TISR)

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_TISR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_TISR field descriptions (continued)

Field	Description
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

48.6.22 LFAST Rx Interrupt Status Register (LFAST_RISR)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RISR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. See "Frames Supported by LFAST interfaces" table for details. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame with payload size for a frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.6.23 LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIISR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received

Table continues on the next page...

LFAST_RIISR field descriptions (continued)

Field	Description
	0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

48.6.24 LFAST PLL and LVDS Status Register (LFAST_PLLLSR)

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved														PLLDIS	PLDCR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LRSLPS	LDSLPS	LDPDS	LRPDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

LFAST_PLLLSR field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0

Table continues on the next page...

LFAST_PLLLSR field descriptions (continued)

Field	Description
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LRSLPS	This bit indicates the real time status of LR sleep signal 0 LR power sleep signal is negated. 1 LR power sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal 0 LD sleep signal is negated. 1 LD sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

48.6.25 LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							URXDV	0					URPCNT		
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRSR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited Data payload present in the LFAST_UNSRDR[8:0].

48.6.26 LFAST Unsolicited Rx Data Register (LFAST_UNSRDR*n*)

Address: 0h base + ACh offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRDR*n* field descriptions

Field	Description
0–31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

48.7 Functional description

48.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

48.7.1.1 LFAST master interface startup procedure

1. After reset the SLCR and RCDCR are programed according to the LVDS parameters in the device data sheet.
2. The PLLCR is programmed with configuration parameters for the PLL.

3. The LCR is programed with configuration parameters of the LVDS.
4. Write MCR[MSEN] = 1. Then select LFAST modes by configuring MCR[CTSEN]and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. Write MCR[RXEN] = 1 and MCR[TXEN] = 1 to negate the LD powerdown, LR disable and LR powerdown signals.
7. Write ICR[ICLCPLD] = 31h to enable the Slaves Tx Interface.
8. Write ICR[SNDICLC] = 1 and ICR[ICLCSEQ] = 1.
9. Write MCR[TXARBD] = 0.
10. The ICLC transmission is confirmed by verifying one of the following:
    - ICR[SNDICLC] = 0.
    - TISR[TXICLCF] = 1.
11. Write ICR[ICLCPLD] = 00h to check the LFAST slaves status.
12. Write ICR[SNDICLC] = 1.
13. The LFAST slave status is confirmed by occurrence of one of the following:
    - LFAST slave is enabled if RIISR[ICPSF] = 1. Proceed to step 14.
    - LFAST slave is disabled if RIISR[ICPFF] = 1. The LFAST master must wait and restart from Step 7.

    Speed mode change:

14. Write PLLCR[SWPON] = 1 to enable the LFAST masters PLL.
15. Write ICLC start PLL frame, ICR[ICLCPLD] = 02h.
16. Write ICR[SNDICLC] = 1.
17. The ICLC transmission is confirmed by occurrence of one of the following:
    - ICR[SNDICLC] = 0.
    - TISR[TXICLCF] = 1.
18. To change LFAST masters Tx interface speed both of the following operations are performed:

Note

(The slaves Rx interface speed should be changed first.)

- Write ICR[ICLCPLD] = 10h for Tx data fast frame.
- Write ICR[SNDICLC] = 1.

19. Both of the following operations are performed to change the LFAST masters Rx interface speed:

Note

(The slaves Tx interface speed mode should be changed first)

- Write ICR[ICLCPLD] = 80h to select Rx data fast frame.
- Write ICR[SNDICLC] = 1.

20. The ICLC transmission is confirmed by the occurrence of one of the following:
    - ICR[SNDICLC] = 0
    - TISR[TXICLCF] = 1.
21. Write SCR[TDR] = 1 to change the LFAST masters Tx interface speed.
22. Write SCR[RDR] = 1 to change the LFAST masters Rx interface speed.
23. Write ICR[ICLCPLD] = 00h to confirm the change in speed of the LFAST slave. This frame should be written after waiting for the expected delay in the start of the PLL and speed mode change delay at the LFAST slave.
24. Write ICR[SNDICLC] = 1
25. Write MCR[TXARBD] = 1, after some delay to ensure the step 24 ICLC frame is send but no other frame is sent to arbitation.
26. The LFAST slaves speed mode is confirmed by the occurrence of the following:
    - If RIISR[ICPSF] = 1. The LFAST slave is in High Speed mode.
    - If RIISR[ICPFF] = 1. The LFAST slave is in Low Speed mode.
27. If RIISR[ICPSF] = 1, then all frame arbitration is enabled by both of the following operations:

- Write ICR[ICLCSEQ] = 0.
- Write MCR[TXARBD] = 0.

28. If RIISR[ICPFF] = 1 modifying the LFAST slaves speed needs to be repeated by:
    - Write SCR[TDR] = 0 to return the Tx interface to Low Speed mode.
    - Write MCR[TXARBD] = 0, to enable frame transmission.
    - The operations from step 14 need to be executed again.

48.7.1.2 LFAST slave interface startup procedure

1. After reset the SLCR and RCDCR are programed according to the LVDS datasheet.
2. The PLLCR is programed with the configuration parameters for the PLL.
3. The LCR is programed with the configuration parameters for the LVDS.
4. Write MCR[MSEN] = 0. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. The LR is enabled by writing MCR[RXEN] = 1. This negates the LR disable signaland LR's power down signal.
7. The LD enable signal needs to be asserted. This is done when one of the following are true:
    - When an ICLC frame with payload 31h is received the H/W will write RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
    - MCRMCR[TXEN] = 1.
8. After a write to MCR[TXARBD] = 0, a ping frame will be sent on one of the following conditions:
    - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h frame is received. H/W writes [ICRPF] = 1.
    - PICR[PNGREQ] = 1.

    Speed mode change:

9. The PLL will start when one of the following conditions is met:

- When an ICLC frame with payload 02h is received. H/W writes RIISR[IPONF] = 1.
- Write PLLCR[SWPON] = 1.

10. The speed of the Tx interface is changed on one of the following conditions:
    - When SCR[DRMD] = 1 and an ICLC frame with payload 80h is received. H/W writes RIISR[ICTFF] = 1 and SCR[TDR] = 1.
    - SCR[TDR] = 1, when SCR[DRMD] = 0.
11. The speed of the Rx interface is changed on one of the following conditions:
    - When SCR[DRMD] = 1 and an ICLC frame with payload 10h is received. H/W writes RIISR[ICRFF] = 1 and SCR[RDR] = 1.
    - SCR[RDR] = 1, when SCR[DRMD] = 0.
12. A Ping Frame is sent on one of the following conditions:
    - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h is received. H/W writes RIISR[ICRPF] = 1.
    - Write PICR[PNGREQ] = 1.
    - Write PICR[PNGREQ] = .

48.7.2 Line Receiver

This section describes the Line Receiver.

48.7.2.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 48-4 shows the block diagram of Uplink Controller.

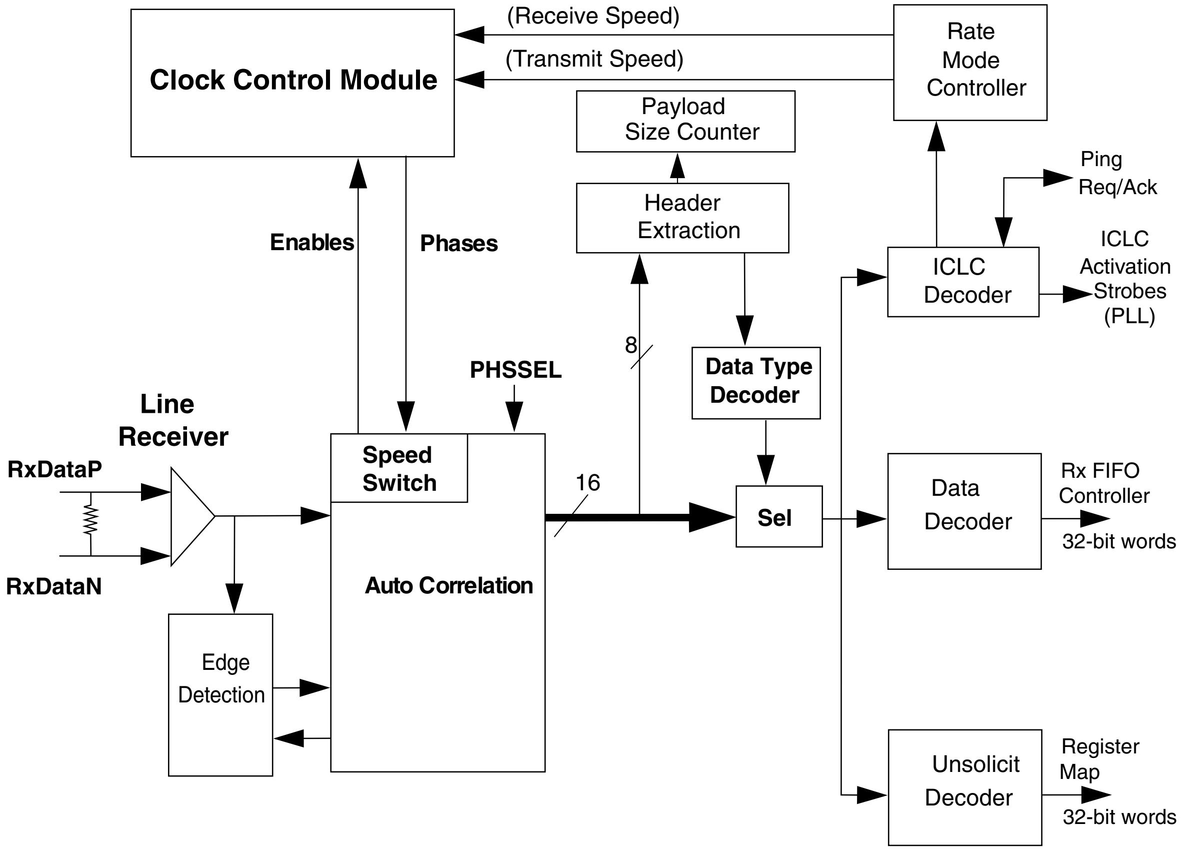


Figure 48-4. Top Level Receive Controller

48.7.2.2 Edge detection and auto-correlation

The edge detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge detection circuitry is used or not. The edge detection will be performed first if required before auto-correlation.

48.7.2.2.1 Auto-Correlation modes

This section describes the Auto-Correlation modes.

48.7.2.2.1.1 Hunt Correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 48-5 shows the block diagram of Hunt Correlation mode.

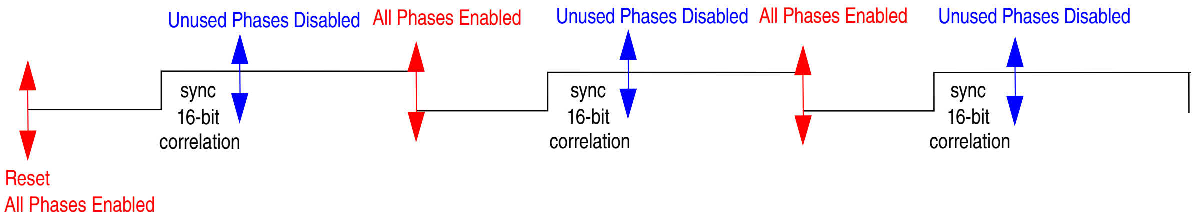


Figure 48-5. Hunt Correlation mode

48.7.2.2.2 Edge Detection

The edge detector is disabled in Hunt Correlation mode.

48.7.2.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in the following figure.

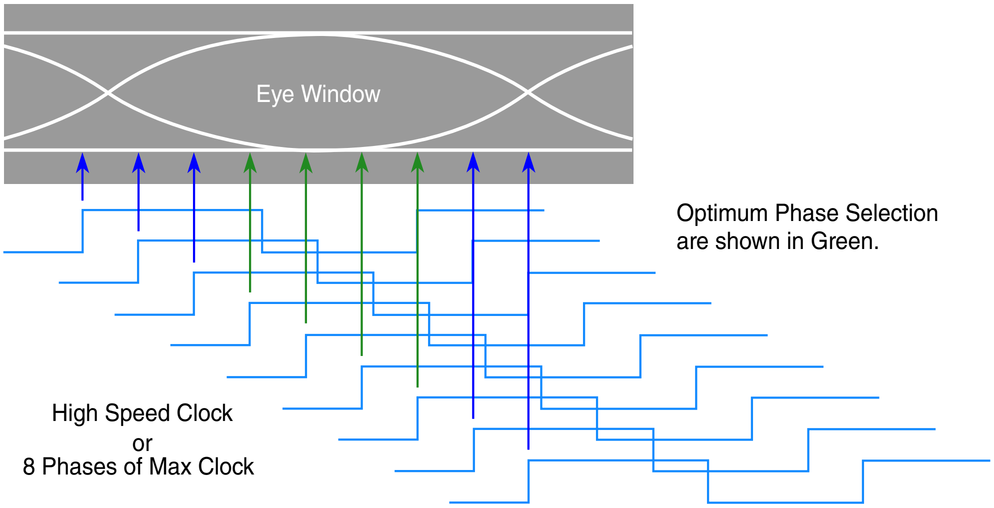

Eye Window
Optimum Phase Selection are shown in Green.
High Speed Clock or 8 Phases of Max Clock

Figure 48-6. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in the following figure.

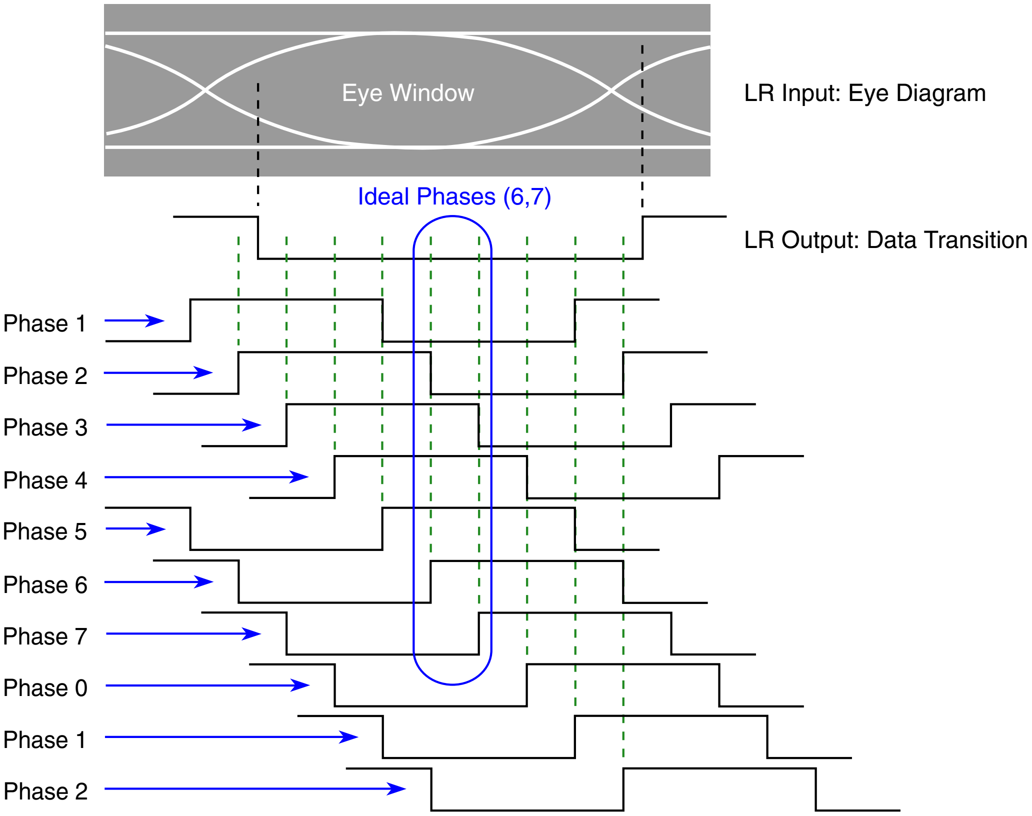


Figure 48-7. High Speed 8 Phase Clock Alignment Example

Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in Figure 48-8 and Figure 48-9.

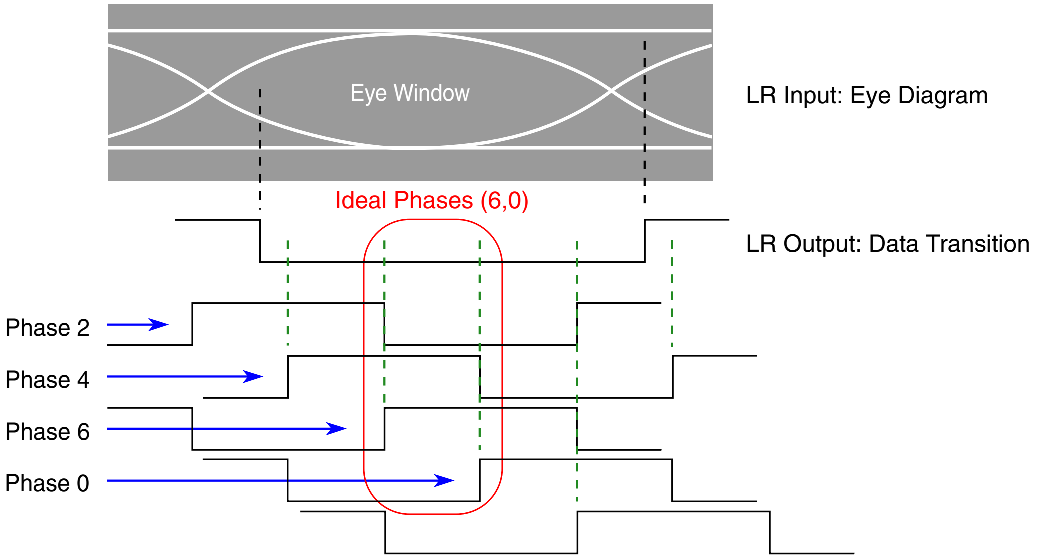


Figure 48-8. High Speed 4 Phase Clock Alignment Example

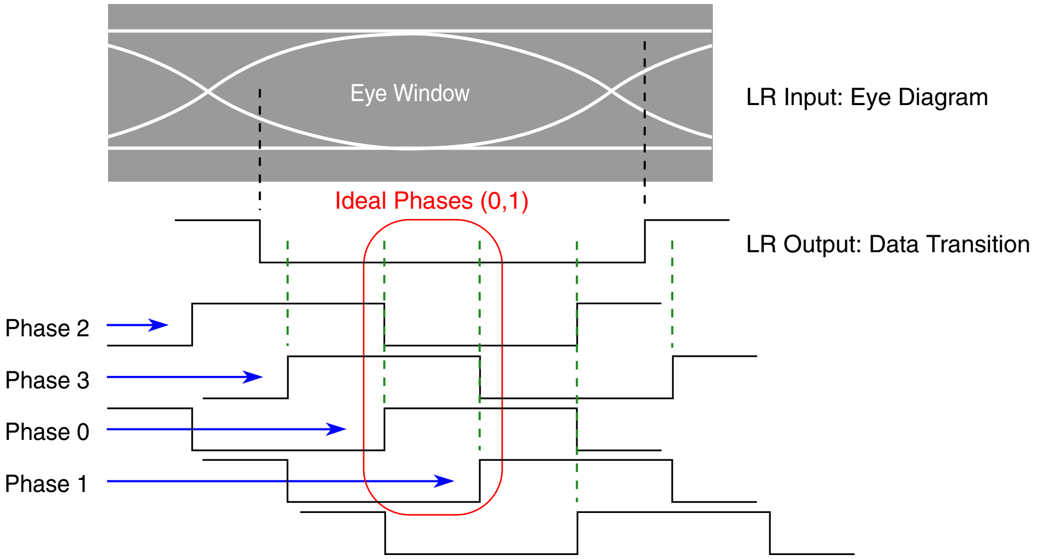


Figure 48-9. Low speed 4 phase clock alignment example

The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High speed 8-phases: Samplers 0,1,2,3,4,5,6,7
- High speed 4-phases: Samplers 0,2,4,6
- Low Speed 4-phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and resamples it to a intermediate phase as shown in Table 48-2, Table 48-3, and Table 48-4 before resampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 48-2. High speed 8 phase selection - sampling procedure

Samplers	InitialSample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 48-3. High speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See Table 48-3, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 48-4. Low speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0

Table continues on the next page...

Table 48-4. Low speed 4 phase selection - sampling procedure (continued)

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See Table 48-4, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

48.7.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore, after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in the following figure.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for example, Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example below, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by deasserting the respective phase enables.

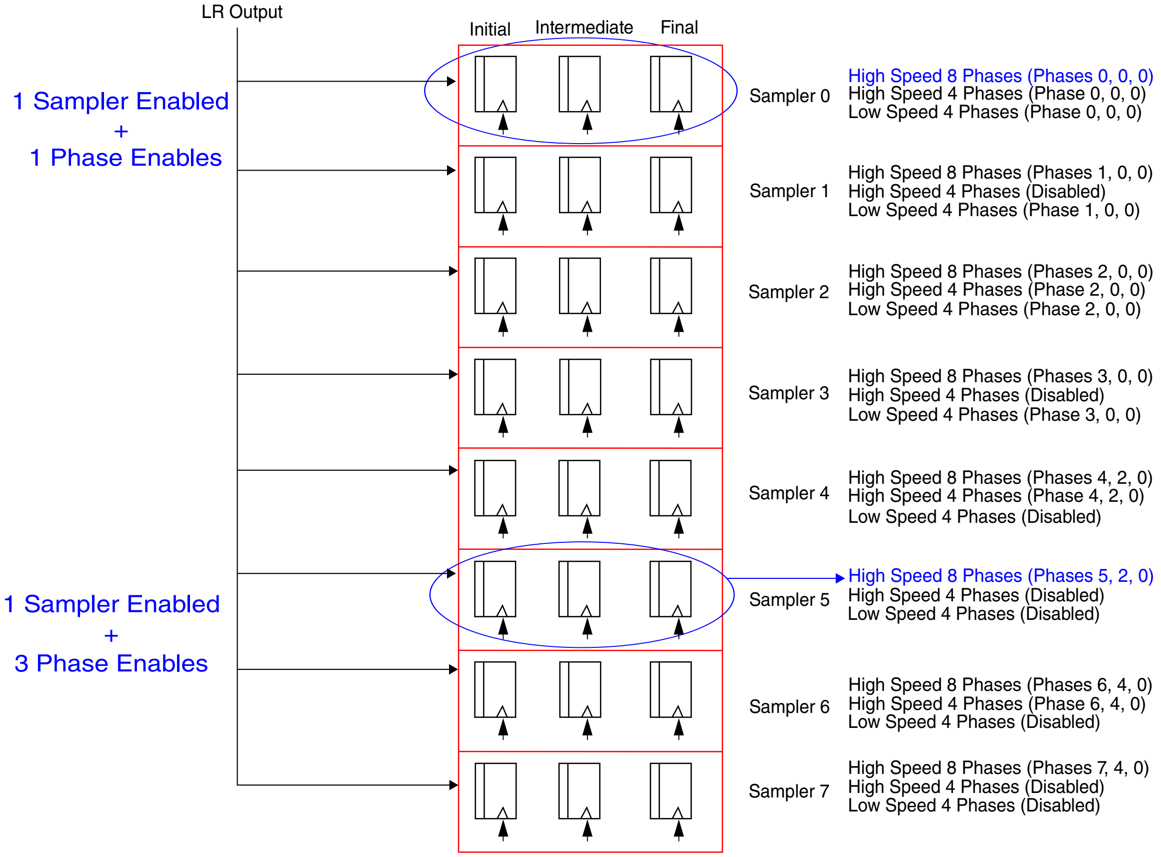

LR Output
Initial
Intermediate
Final
1 Sampler Enabled
+
1 Phase Enables
1 Sampler Enabled
+
3 Phase Enables
Sampler 0
High Speed 8 Phases (Phases 0, 0, 0)
High Speed 4 Phases (Phase 0, 0, 0)
Low Speed 4 Phases (Phase 0, 0, 0)
Sampler 1
High Speed 8 Phases (Phases 1, 0, 0)
High Speed 4 Phases (Disabled)
Low Speed 4 Phases (Phase 1, 0, 0)
Sampler 2
High Speed 8 Phases (Phases 2, 0, 0)
High Speed 4 Phases (Phase 2, 0, 0)
Low Speed 4 Phases (Phase 2, 0, 0)
Sampler 3
High Speed 8 Phases (Phases 3, 0, 0)
High Speed 4 Phases (Disabled)
Low Speed 4 Phases (Phase 3, 0, 0)
Sampler 4
High Speed 8 Phases (Phases 4, 2, 0)
High Speed 4 Phases (Phase 4, 2, 0)
Low Speed 4 Phases (Disabled)
Sampler 5
High Speed 8 Phases (Phases 5, 2, 0)
High Speed 4 Phases (Disabled)
Low Speed 4 Phases (Disabled)
Sampler 6
High Speed 8 Phases (Phases 6, 4, 0)
High Speed 4 Phases (Phase 6, 4, 0)
Low Speed 4 Phases (Disabled)
Sampler 7
High Speed 8 Phases (Phases 7, 4, 0)
High Speed 4 Phases (Disabled)
Low Speed 4 Phases (Disabled)

Figure 48-10. 8 Samplers, each Sampler has 3 Registers

In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in the following figure.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

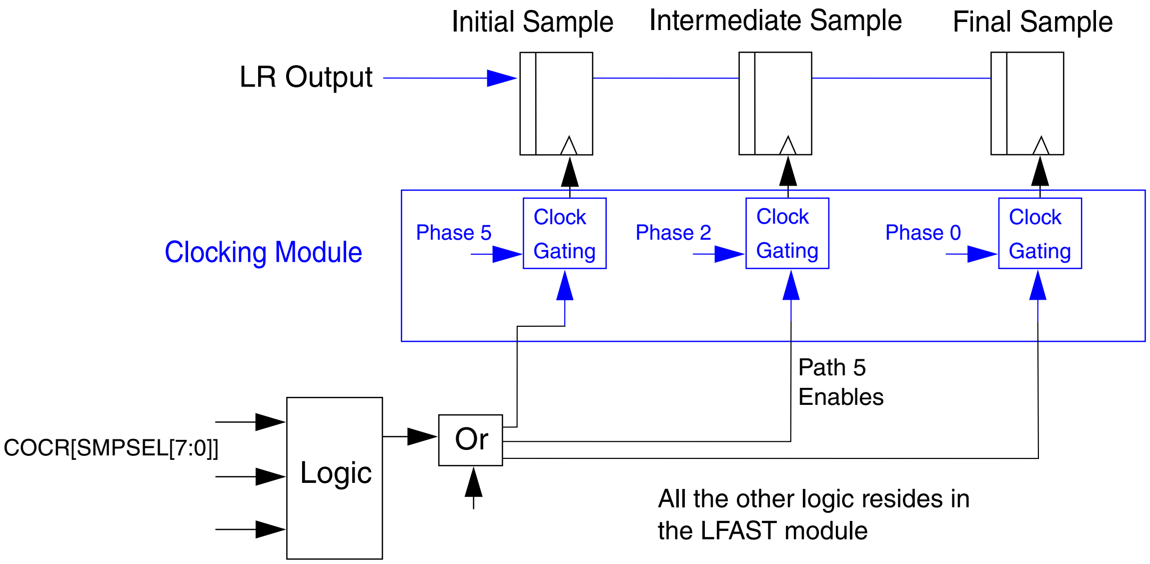


Figure 48-11. Sampler 5 Logic

48.7.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in the Figure 48-12.

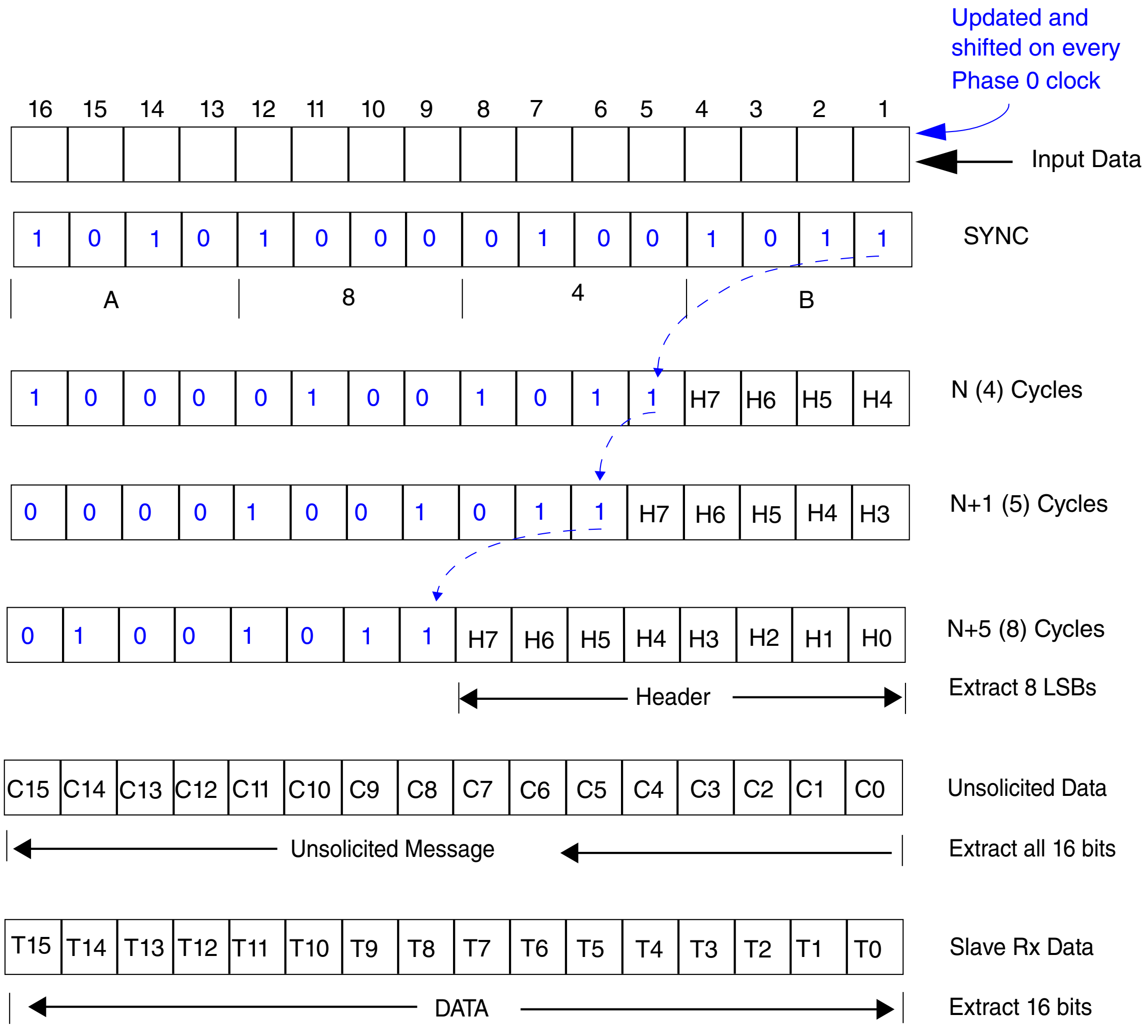

Updated and shifted on every Phase 0 clock
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Input Data
1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1
SYNC
A
8
4
B
1 0 0 0 0 1 0 0 1 0 1 1 H7 H6 H5 H4
N (4) Cycles
0 0 0 0 1 0 0 1 0 1 1 H7 H6 H5 H4 H3
N+1 (5) Cycles
0 1 0 0 1 0 1 1 H7 H6 H5 H4 H3 H2 H1 H0
N+5 (8) Cycles
Header
Extract 8 LSBs
C15 C14 C13 C12 C11 C10 C9 C8 C7 C6 C5 C4 C3 C2 C1 C0
Unsolicited Data
Unsolicited Message
Extract all 16 bits
T15 T14 T13 T12 T11 T10 T9 T8 T7 T6 T5 T4 T3 T2 T1 T0
Slave Rx Data
DATA
Extract 16 bits

Figure 48-12. Extracting Header and Payload from the 16-bit output from Auto correlation.

48.7.3 Transmit Controller

This section describes the Transmit Controller.

48.7.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, CTS, ping, and so on. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in the following figure.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

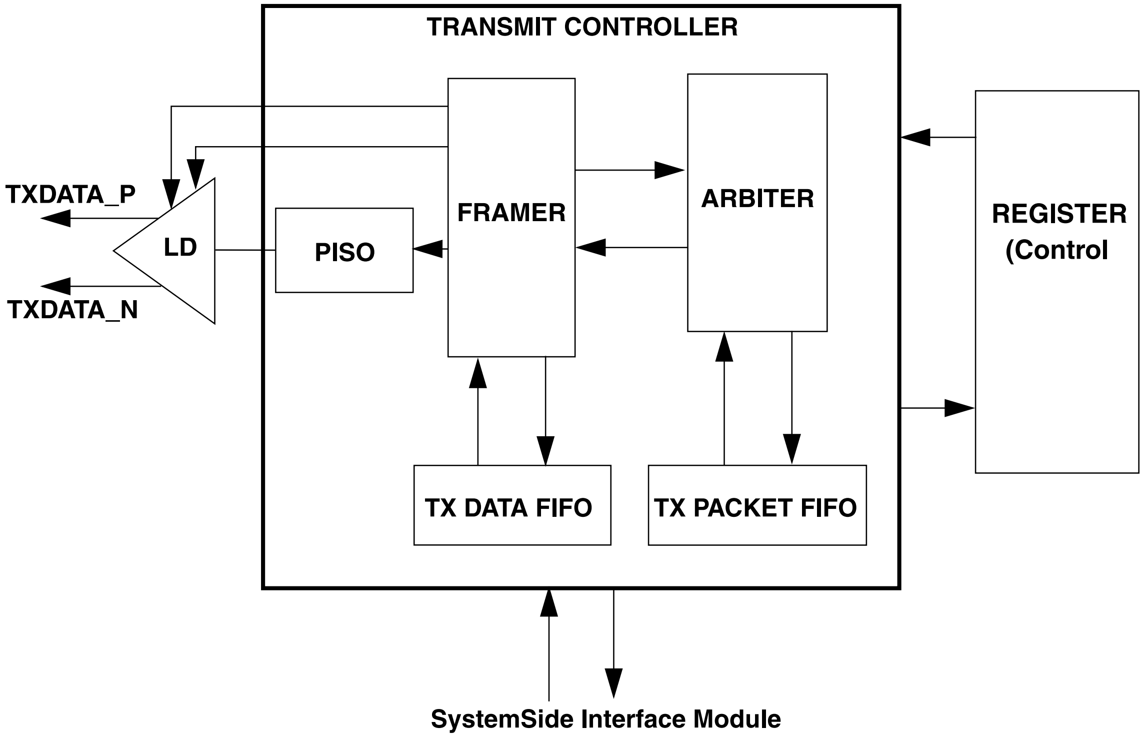


Figure 48-13. Transmit Controller Connections

48.7.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like

- S/W programmable registers
- System side interface FIFO
- Rx interface controller

The level of priority for each request is shown in Table 48-5

Table 48-5. Priority Levels for the Transmit Controller

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
LFAST interface enable	1	1	• LFAST interface enable is asserted • LFAST interface enable is negated
Tx Interface disabled	2	2	• MCR[DRFEN] = 0 • MCR[TXEN] = 0 • MCR[DRFRST] = 1 • MCR[TXARBD] = 1 • LFAST Slave: ICLC frame with payload for "Disable Rx interface" received
Tx Interface speedmode change	3	3	• SCR[TDR] is modified • LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	4	• TMCR[LPON] = 1Dig • RF Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
ICLC frame request	5	-	• LFAST Master: ICR[SNDICLC] = 1 • LFAST Master: ICR[ICLCSEQ] = 1
Ping response request	-	5	• LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 • LFAST Slave: PICR[PNGREQ] = 1

Table continues on the next page...

Table 48-5. Priority Levels for the Transmit Controller (continued)

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
Unsolicited frame request	6	6	• Last Frame with CTS = 1 received from the peer LFAST device • UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	7	• Tx FIFO contains one or more frames • Last Frame with CTS = 1 received from the peer LFAST device • MCR[DATAEN] = 1
CTSFrame	8	8	• MCR[CTSEN] = 1 • Rx FIFO reaches High/Low threshold, defined by TISR and no frame pending
Clock mode test	9	9	• TMCR[CLKTST] = 0 • LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

48.7.3.3 Line Driver digital connections

48.7.3.3.1 Line Driver states

The LD has the following states:

- Shutdown

The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling time is required before the LD may be used for communication.The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

  The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In the sleep state the LD outputs are connected to Vcm (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a pre-defined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

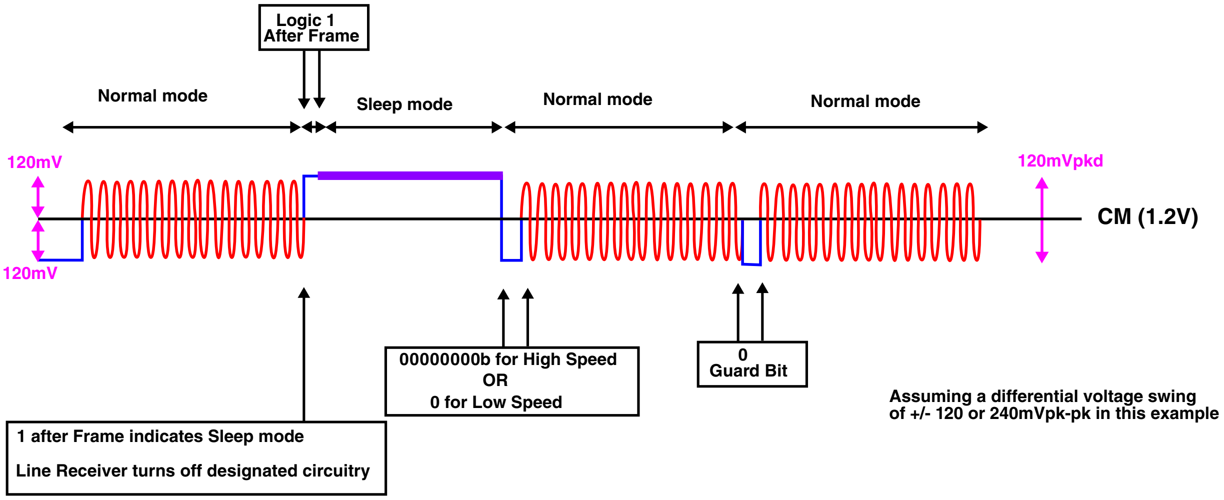

Logic 1
After Frame
Normal mode
Sleep mode
Normal mode
Normal mode
120mV
120mV
120mVpkd
CM (1.2V)
00000000b for High Speed
OR
0 for Low Speed
0
Guard Bit
Assuming a differential voltage swing
of +/- 120 or 240mVpk-pk in this example
1 after Frame indicates Sleep mode
Line Receiver turns off designated circuitry

Figure 48-14. Example of Sleep mode

As shown in Figure 48-14 before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

  In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. If the LD is not enabled, and the framer is activated, and has a frame to transmit, an interrupt (TISR[TXIEF]) will be generated. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 48-6. Line Driver States

LD State	LFAST Master Triggers	LFAST Slave Triggers
Shutdown	• Programing LVDS[SWOFFLD] and LVDS[SWONLD] • Programing MCR[TXEN] and MCR[DRFEN]	• LFAST interface enable negation/assertion • Programing LVDS[SWOFFLD] and LVDS[SWONLD] • ICLC command from LFAST master • Programing MCR[TXEN] and MCR[DRFEN]
Sleep	• Programing LVDS[SWSLPLD] and LVDS[SWWKLD]	• No pending frame for transmission • Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

48.7.4 CTS mode support

LFAST module supports flow control methods. The CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by proper programing of the Rx FIFO Lower threshold (RFCR[CTSMN]) and Higher threshold (RFCR[CTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by write MCR[CTSEN] = 1 then:

- Sends CTS bit as 0 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield TISR[CTSMX]. This indicates that the LFAST device doesn't want the peer device to send data. The CTS bit of all frame are sent 0 untill the Rx FIFO pointer reaches the lower threshold, as described below.

- Sends CTS bit as 1 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield TISR[CTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frame are sent 1 untill the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of Header field are sent as 0011b and 8-bit payload of 0.

If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frame sent will have CTS bit as 1 in LFAST frame (bit[0] of Header field).

CTS Reception:

The reception of a LFAST frame with CTS bit 0 indicates that the peer device is not ready to receive data frames. The transmission of all pending data and unsolicited frames are postponed untill a frame with CTS bit 1 is received.

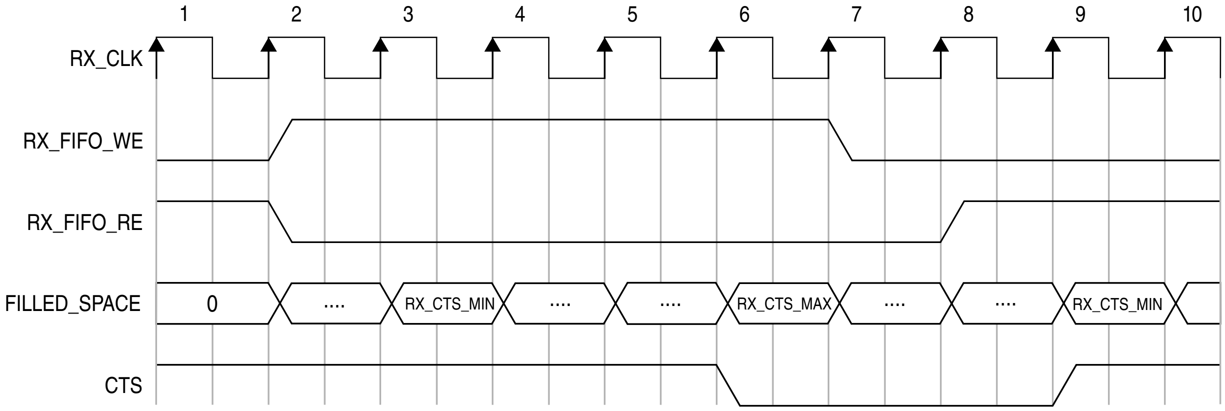


Figure 48-15. CTS generation

48.7.5 Frames supported

Table 48-7 provides the frames supported and their permissible payload sizes

Table 48-7. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx2. LFAST Slave Rx	Supported by1. LFAST Slave Tx2. LFAST Master Rx	Payload Size (bits)
Data Frame	0100b – 1011b	YES	YES	32, 64, 96, 128, 256, 288
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO[1]	8
CTS Frame	0011b	YES	YES	8
Reserved	All others	—	—	—

[1] Except the ICLC PING response frame.

If logical channel type (LCT) code other than the above mentioned are received then the interrupt flag RISR[RXLCEF] = 1. If the payload size received does not match any value in Table 48-7 then the flag RISR[RXSZF] = 1. In both these error conditions the received frame is ignored. The S/W may have to take the necessary steps to recover from such an error.

48.7.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

48.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

48.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

48.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer
5. CTS is set depending on the push/pull mode setting defined by MCR[CTSEN].

48.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded. In this case, UNSRSR[RXOF] = 1 and an interrupt is generated if the RIER[RXOFIE] = 1.

48.7.6.2 Unsolicited flow

48.7.6.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

48.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
    - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
        - UNSTCR[USNDRQ] = 0
        - TISR[TXUNSF] = 1
    - If UNSTCR[USNDRQ] = 0 then:
        - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
        - Program the unsolicited frame header in UNSTCR[UNSHDR].
        - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.
6. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

48.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.

3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.
3. Then it clears the interrupt (write RISR[RXUNSF] = 1) and also the UNSRSR[URXDV] bit.

48.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

48.7.6.3.1 ICLC data transmit flow

Whenever the processor intends to transfer an ICLC frame to the LFAST slave then the following steps are to be followed.

1. Write the ICLC frame payload in the ICR[ICLCPLD] (see Table 48-8 for payloads as defined by the standard).
2. Program ICR[SNDICLC] = 1 to initiate the ICLC frame transfer. This bit clears itself when the ICLC frame has been transmitted. The processor needs to poll this bit to make sure it is cleared before setting this bit again (even when ICR[ICLCSEQ] = 1).
3. TISR[TXICLCF] = 1 after the transfer of the ICLC frame.
4. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

To determine whether the ICLC frame has been sent or not, either the ICR[SNDICLC] bit can be polled, or wait for TISR[TXICLCF] = 1.

Table 48-8. Supported ICLC Payloads

ICLC Code(hex)	ICLC Function
00	Ping Request from LFAST Master to LFAST Slave
01	Reserved
02	Start PLLIn preparation for High Speed mode
04	Stop PLL Fallback to low speed mode
08	Select TxData Slow (LFAST Slave Rx Interface)
10	Select TxData Fast (LFAST Slave Rx Interface)
20	Select RxData Slow (LFAST Slave Tx Interface)
40	Not Supported
80	Select RxData Fast (LFAST Slave Tx Interface)
31	Enable RxData Interface (LFAST Slave Tx Interface)
32	Disable RxData Interface (LFAST Slave Tx Interface)
34	Clock Test mode Send 101010… continuously using the currently configured clock rate (slow, fast);
FF	Turn payload loopback on
38	Turn Test mode (Loopback and Clock Test) off

Programing model for ICLC frame transmit:

1. Set the ICR[ICLCSEQ] bit (optional)
2. Write the ICLC frame payload in the ICR[ICLCPLD], corresponding to the desired Tx/Rx data rate change, as described in the Table 48-8.
3. Write ICR[SNDICLC] = 1 to initiate the ICLC frame transfer.
4. LFAST master will schedule the ICLC transfer. Data present in ICR[ICLCPLD] is transmitted to the LFAST slave at the old Tx data rate. LFAST slave will configure its Tx/Rx interface data rate accordingly.
5. Reset the ICR[ICLCSEQ] to allow the scheduling of other types of frames, which will be sent at the new data rate.

When ICR[ICLCSEQ] = 0, though the ICLC frame transfer still has highest arbitration priority, other frames may be scheduled if no valid ICLC frame request exist.

Note

ICLC frame transmit will not trigger the internal data rate change of LFAST Master Tx/Rx interface. LFAST master Tx/Rx interface data rate will only be changed, when corresponding SCR[TDR] and SCR[RDR] bits are appropriately configured.

Processor requires to take care that no data frames are to be transmitted while data rate change is happening or ICLC is transmitting frames by setting ICR[ICLCSEQ] bit in Table 48-8.

48.7.6.3.2 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

The reception of an ICLC frame is indicated by an interrupt to the system. The ICLC status register RISR indicates the type of the ICLC frame received. The Rx block will decode and generate the appropriate signals. An invalid ICLC code besides the ones listed in Table 48-8 will cause RISR[RXICF] = 1.

48.7.6.3.2.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF]. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

48.7.6.3.2.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

48.7.6.3.2.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

48.7.6.3.2.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

48.7.6.3.2.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

48.7.6.3.2.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

48.7.6.3.2.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable.

48.7.6.3.2.8 Ping response ICLC

The LFAST master considers any ICLC frame payload as ping response data. The LFAST masters Rx block will store the received ping data in PISR[RXPNGD] and match it with PICR[PNGPYLD]. RIISR[ICPSF] = 1 if the Ping received matches with the one stored in PICR[PNGPYLD], or RIISR[ICPFF] = 1.

48.7.6.4 CTS flow

48.7.6.4.1 CTS Tx flow

The CTS frame, if enabled by the S/W, indicates the status of the Receive Data FIFO. CTS frame is triggered whenever Rx Data FIFO reaches either the High/Low threshold and no data, ICLC or unsolicited frame is ready for transmission.

Programing model for CTS frame transmit:

- Program RFCR[CTSMX] and RFCR[CTSMN].
- Enable the CTS frame Transmission by programming MCR[CTSEN] = 1.
- Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path.
- The CTS bit sent with a valid frame (for example, data, ICLC and unsolicited) is 1 until the Rx Data FIFO reaches High Threshold.
- When the Rx Data FIFO reaches higher threshold and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 0 and the status bit RISR[RXMXF] = 1.
- When the Rx Data FIFO reaches Low Threshold, due to system side reads and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 1, and status bit RISR[RXMNF] = 1.

48.7.6.4.2 CTS Rx flow

Whenever a CTS frame header or any other frame with valid header is received with CTS bit = 0, then RISR[RXCTSF] = 1. The Tx Interface arbitration for unsolicited frame and data frame transmit request is turned off on reception of frame with CTS bit 0. The arbitration for unsolicited frame and data frame is enabled on reception of frame with CTS bit 1. Frame with LCT type CTS are not stored in the LFAST.

48.7.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

48.7.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 48-9. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.

- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

48.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. CTS bit (Bit 0) of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

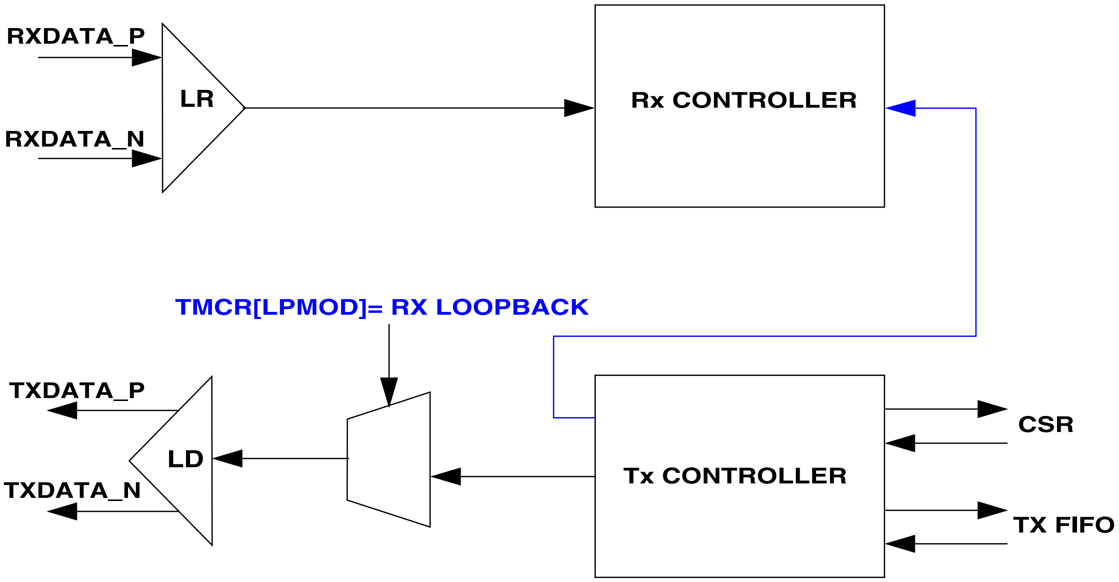


Figure 48-16. Rx LoopBack mode

Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

48.7.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via "Rx LVDS LoopBack Mux". Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

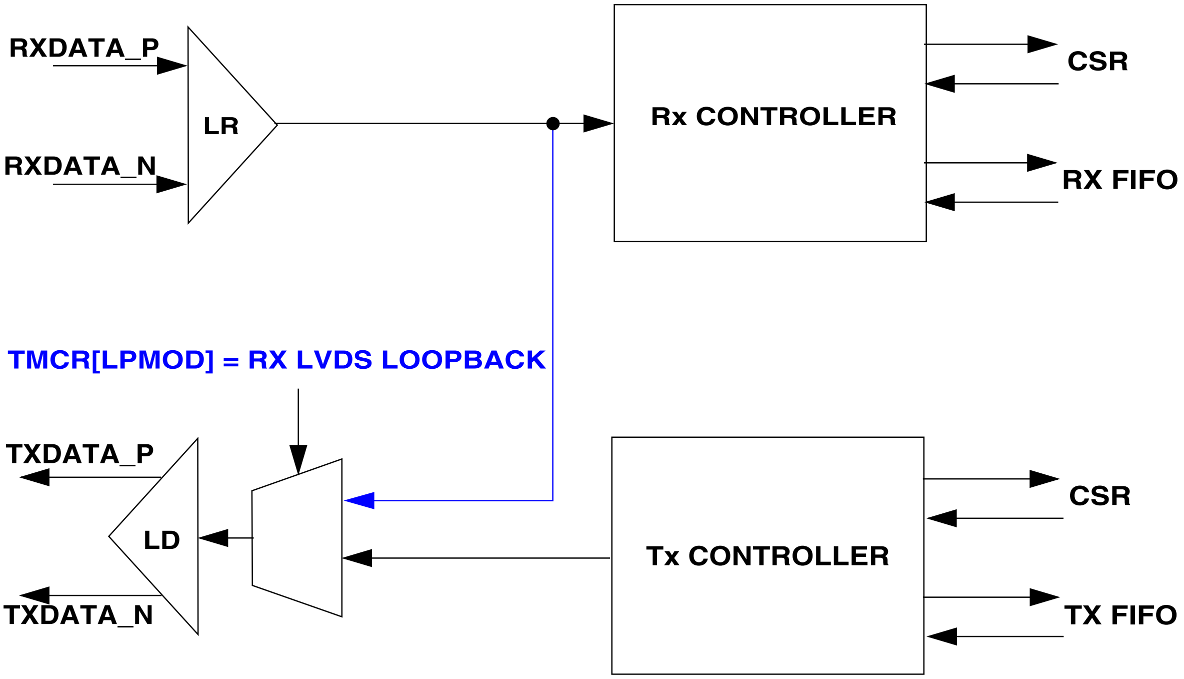


Figure 48-17. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:

  - S/W writes TMCR[LPON] = 1.
  - For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
  - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.

48.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

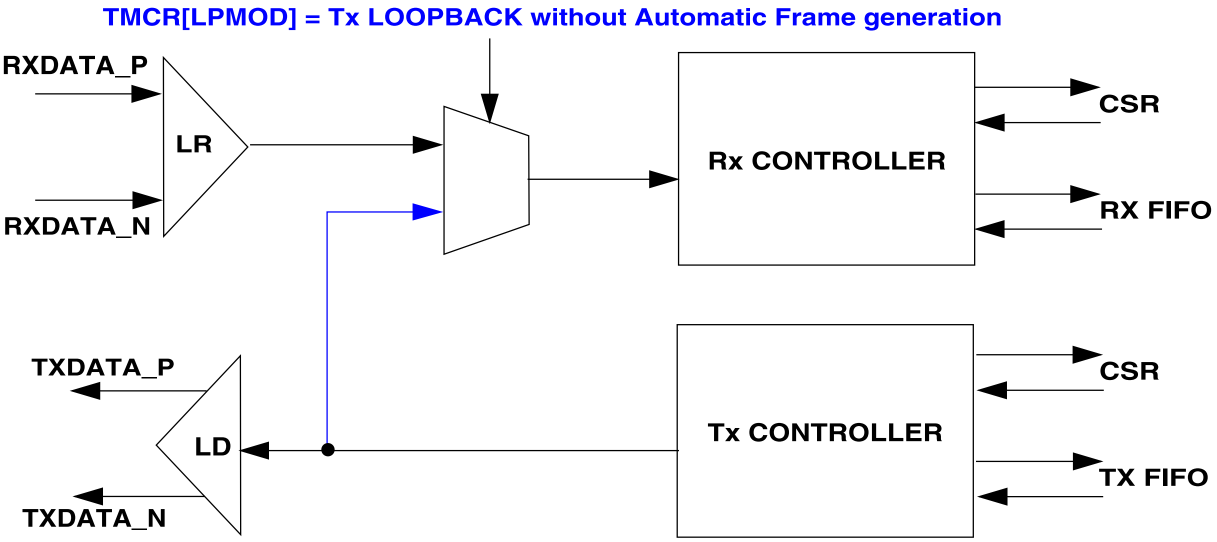


Figure 48-18. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).

All frames and error flags are decoded in this mode.

48.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where pre-defined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

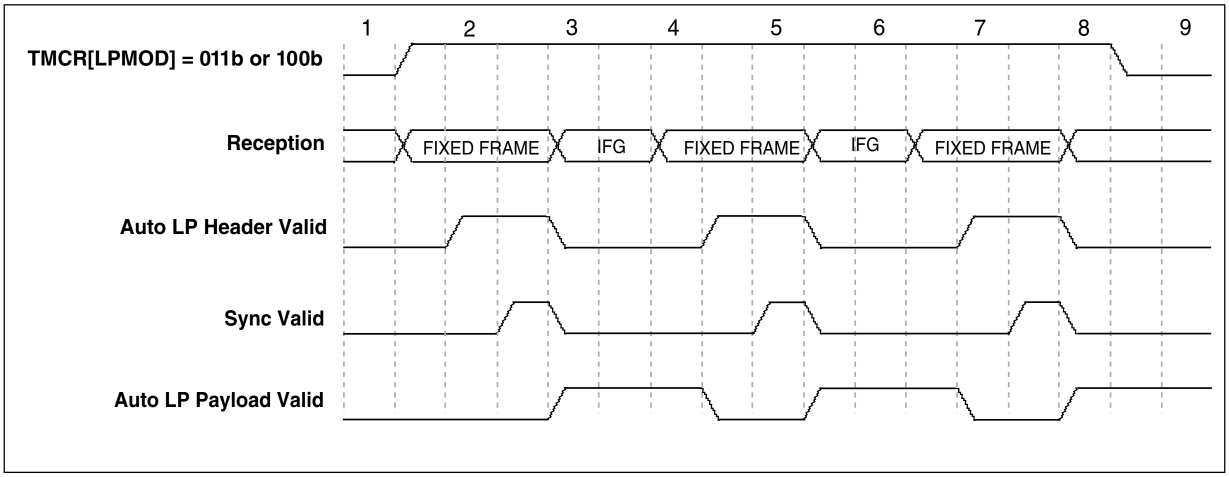


Figure 48-19. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

1. Tx Loopback with Automatic frame generation
2. Tx LVDS (external) with Automatic frame generation

48.7.7.1.4.1 Tx loopback mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

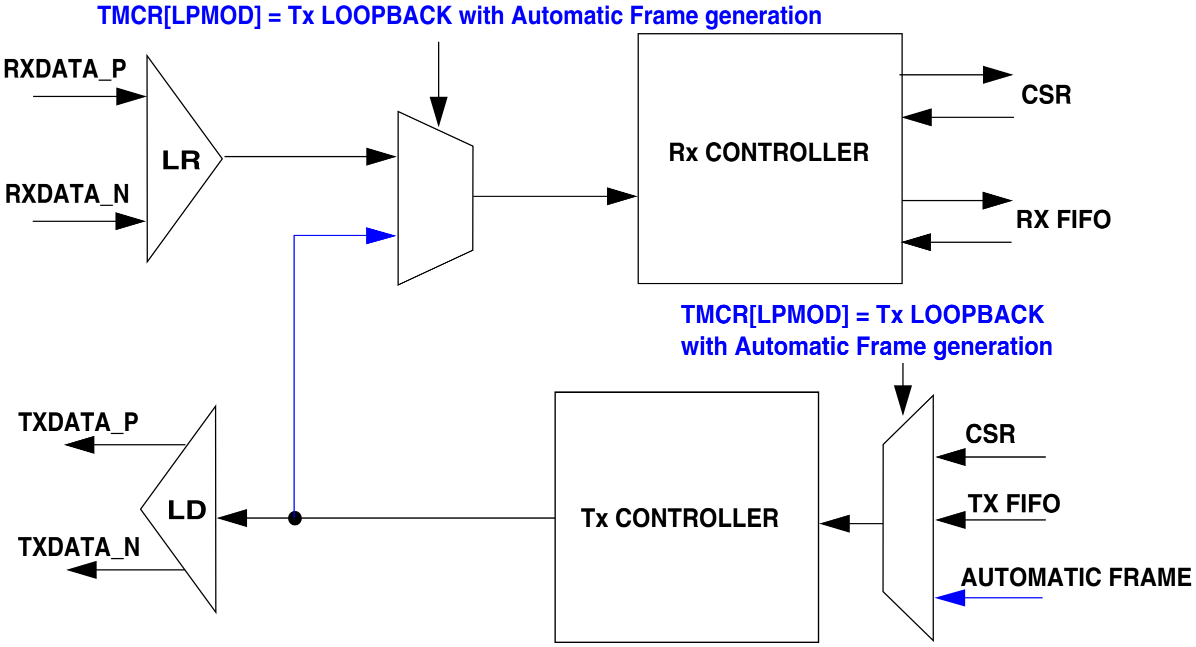

TMCR[LPMOD] = Tx LOOPBACK with Automatic Frame generation
RXDATA_P
RXDATA_N
LR
Rx CONTROLLER
CSR
RX FIFO
TMCR[LPMOD] = Tx LOOPBACK
with Automatic Frame generation
TXDATA_P
TXDATA_N
LD
Tx CONTROLLER
CSR
TX FIFO
AUTOMATIC FRAME

Figure 48-20. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
    - S/W programs TMCR[LPON] = 1.
    - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.

48.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

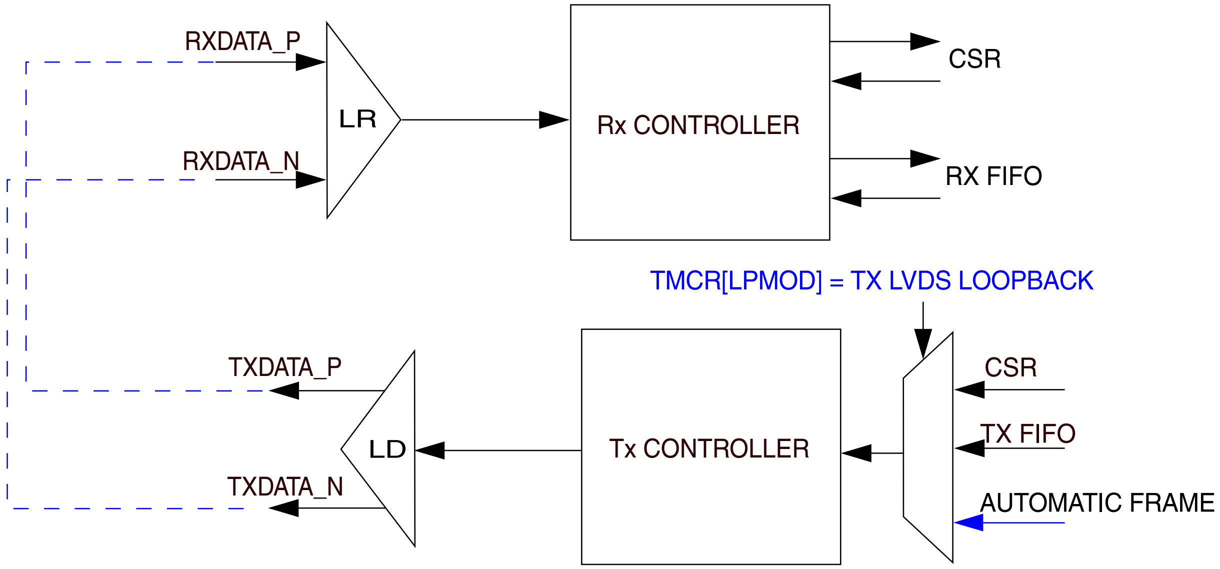


Figure 48-21. Tx LVDS loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
    - S/W programs TMCR[LPON] = 1.
    - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx LVDS loopback with automatic frame generation mode can be done by the following method:

- S/W programs TMCR[LPON] = 0.

48.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by either of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode doesn't affect the functionality of the Rx Interface.

48.7.8 Interrupts

The LFAST module generates five interrupts to the processor:

- Transmit complete interrupt
- Transmit exception interrupt
- Receive complete interrupt
- Receive exception interrupt
- ICLC recieve interrupt

Each interrupt is asserted when any one of their status conditions is met.

48.7.8.1 Transmit complete interrupt

This interrupt indicates a transfer compeletion of a frame. This interrupt is asserted whenever any of the following bits are set in the TISR and also the corresponding mask bits are set in the TIER.

- TXDTF
- TXICLCF
- TXUNSF
- TXPNGF

Each of these bits can be individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

48.7.8.2 Transmit exception interrupt

This interrupt indicates occurrence of an exception condition in the Tx block. This interrupt is asserted whenever any of the following bits are set in the TISR along with the corresponding mask bits in the TIER.

- TXOVF
- TXIEF

Each of these bits can be individually masked. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

Table 48-10. Recommended transmit exception handling

Exception	Recommendation
TXOVF	• MCR[TXEN] = 0 • The system level interfaces transmit is reseted and enabled again • For Slave: After reception of ICLC Ping Response Request the MCR[TXEN] may be set • For Master: The MCR[TXEN] should be set and an ICLC frame Ping Response Request should be sent
TXIEF	• The MCR[TXEN] should be set

48.7.8.3 Receive complete interrupt

This interrupt indicates a reception of a frame. This interrupt is asserted whenever any of the following bits is set in the RISR along with the corresponding interrupt enable bit is also set in the RIER.

- RXCTSF
- RXDF
- RXUNSF

Each of these bits are individually maskable. So, the bits that are not required to generate an interrupt to the processor can be masked by clearing its bit in the RIER. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

48.7.8.4 Receive exception interrupt

This interrupt indicates occurrence of an exception condition in Rx block. This interrupt is asserted whenever any of the following bits is set in the RISR and corresponding enable mask bit is also set in the RIER.

- RXLCEF
- RXICF
- RXSZF
- RXUOF
- RXUFF
- RXMXF
- RXMNF
- RXOFF

Each of these bits are individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked by clearing the appropriate bit in the RIER. For details on the cause of assertion of each bit refer to the RISR description. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

Table 48-11. Recommended receive exception handling mechanism

Exception	Recommendation
RXLCEF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXICF	No action required
RXSZF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXOFF	• The MCR[RXEN] may be cleared • The system level interfaces receive may be reset and enabled, as current transfer may be corrupted
RXUFF	No action required
RXMXF	The system level interfaces receive should be enabled, if not enabled
RXMNF	No action required
RXUNSF	No action required

48.7.8.5 ICLC receive interrupt

This interrupt indicates reception of a valid ICLC frame. This interrupt is asserted whenever any of the following bits is set in the RIISR along with the corresponding enable bit mask bit is set in the RIIER.

- ICPONF
- ICPOFF
- ICTSF
- ICTFF
- ICRSF
- ICRFF
- ICTEF
- ICTDF
- ICCTF
- ICLPF
- ICTOF
- ICPRF
- ICPSF
- ICPFF

Each of these bits are individually maskable. So those bits that are not required to generate an interrupt to the processor can be masked. For details on the cause of assertion of each bit, refer RIISR. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in RIISR.

48.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 48-12. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer(in bits)	Rx Buffer(in bits)	Memory Type
Data Frame	38 × 32 max 6 packets	38 × 32 max 6 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	1 × 7 max 1 packet[1]	1 × 8 max 1 packet[2]	Registers ICR, PISR
CTS Frame	N/A	N/A	N/A

[1] Only for MCR[MSEN] = 1

[2] Only for ping response data when MCR[MSEN] = 1

48.9 Resets

The various blocks in LFAST are reset as described in the table below.

Table 48-13. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON] • ICR[ICLCSEQ] and ICR[SNDICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ]

48.10 Clocks

The LFAST mainly works on three clocks:

- System Bus Clock used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

48.10.1 Clocking strategy

The following figure shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

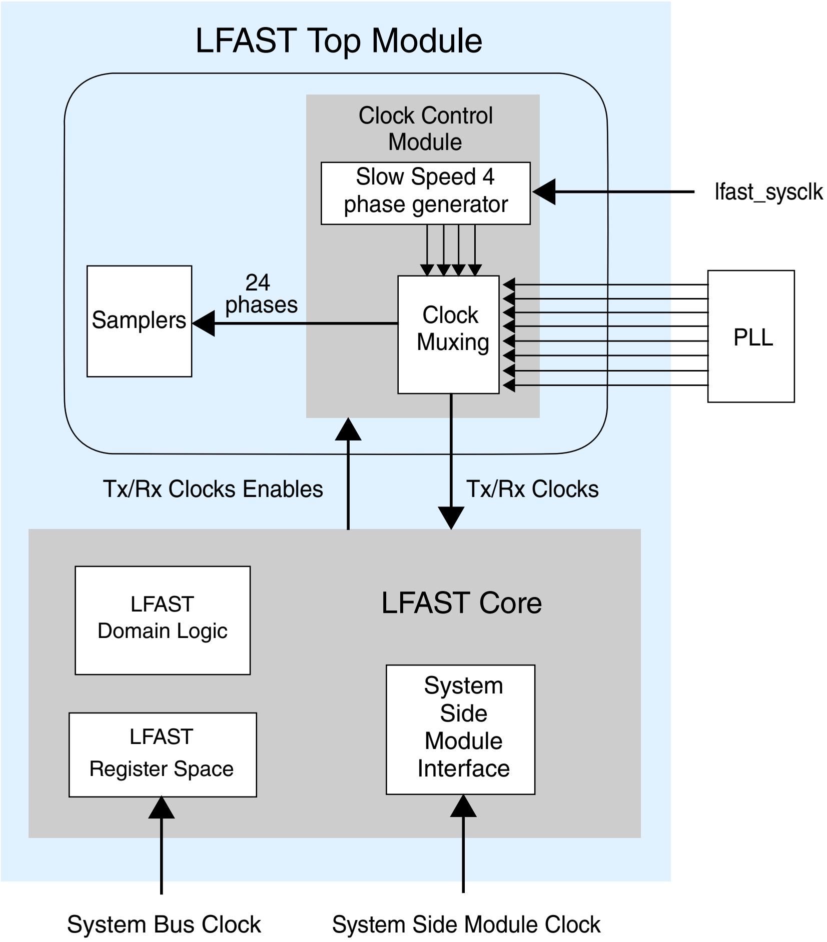


Figure 48-22. LFAST module clock domains

48.10.2 Slow speed clock

48.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in Figure 48-23, Figure 48-24 and Figure 48-25.

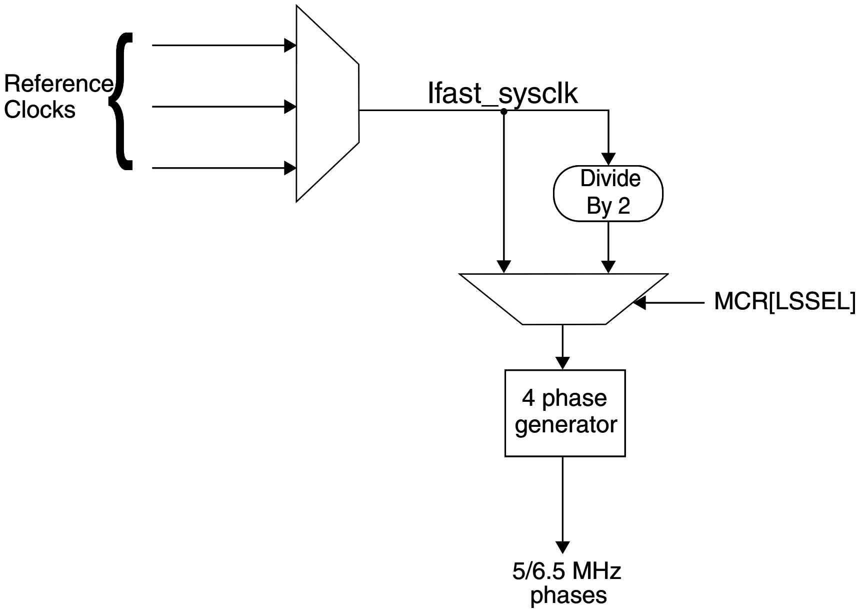


Figure 48-23. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in Figure 48-23.

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 48-24 (selected clock path shown in green).

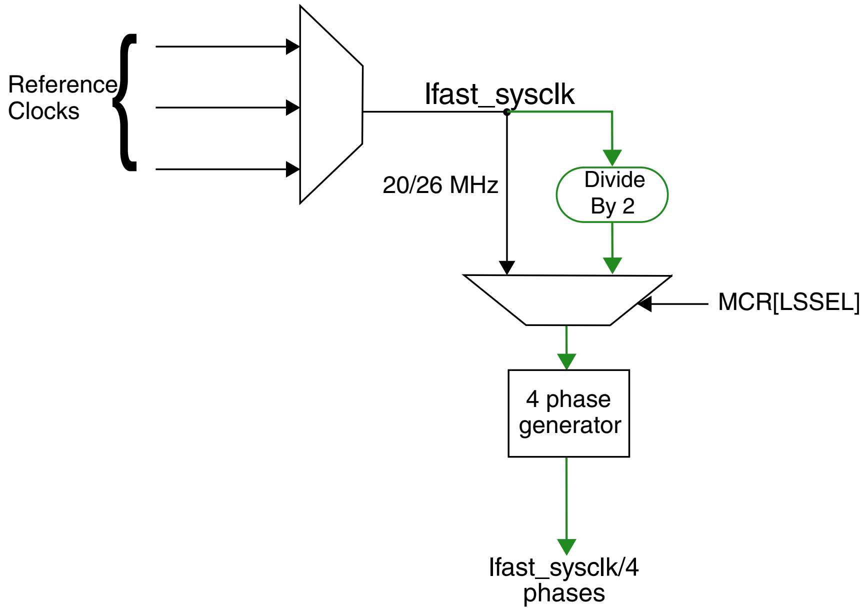


Figure 48-24. External clock muxing of lfast_sysclk of 20/26 MHz

In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 48-25 (selected clock path selected shown in magenta).

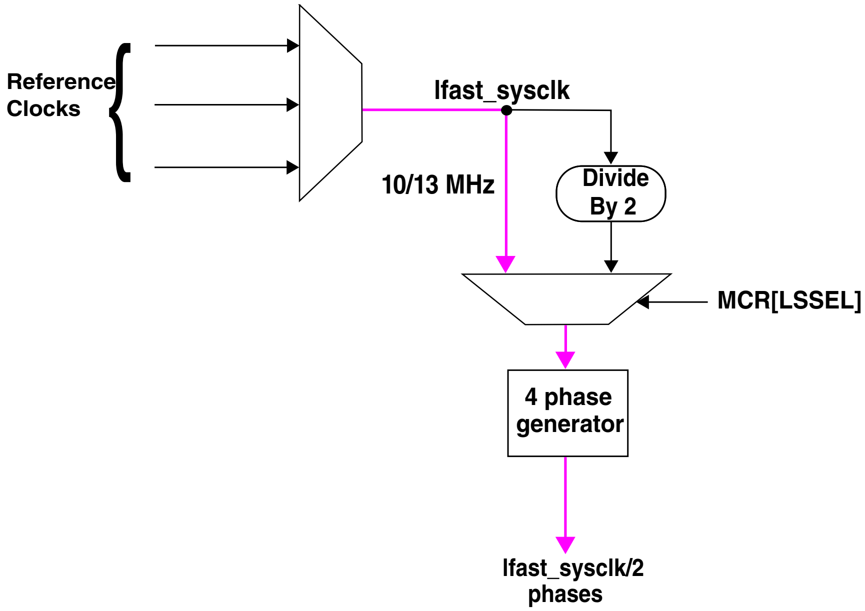


Figure 48-25. External clock muxing of lfast_sysclk of 10 MHz

48.10.2.2 Slow Speed 4 phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. The following figure shows 4 phases getting generated using lfast_sysclk.

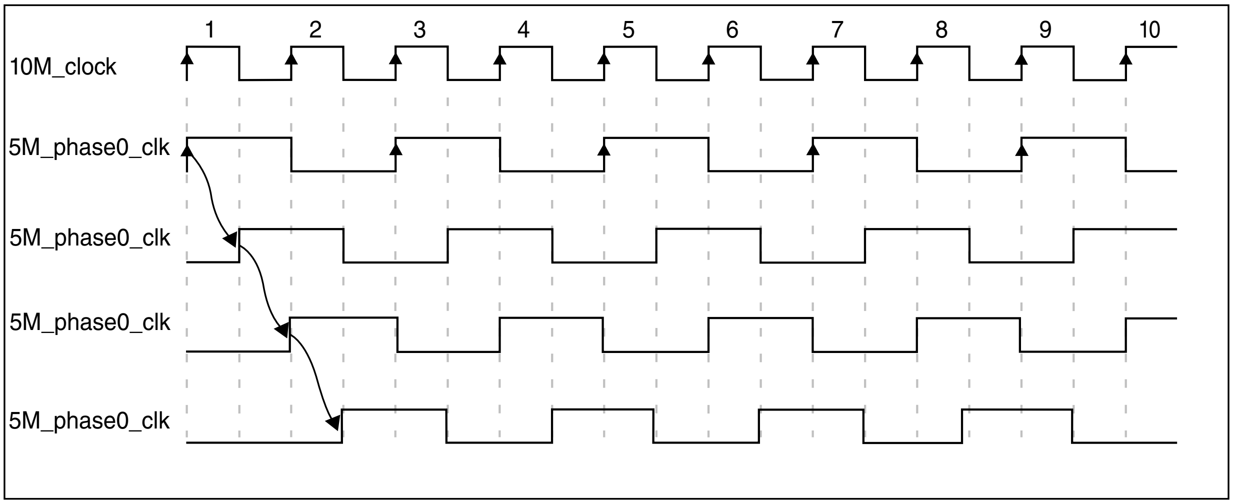


Figure 48-26. Slow speed 4 phases generation

48.10.3 Rx Controller Clocks

48.10.4 Clocking Module Requirements for High Speed Phases

The high speed phases are obtained from the PLL via the Clocking Module as shown in the following figure.

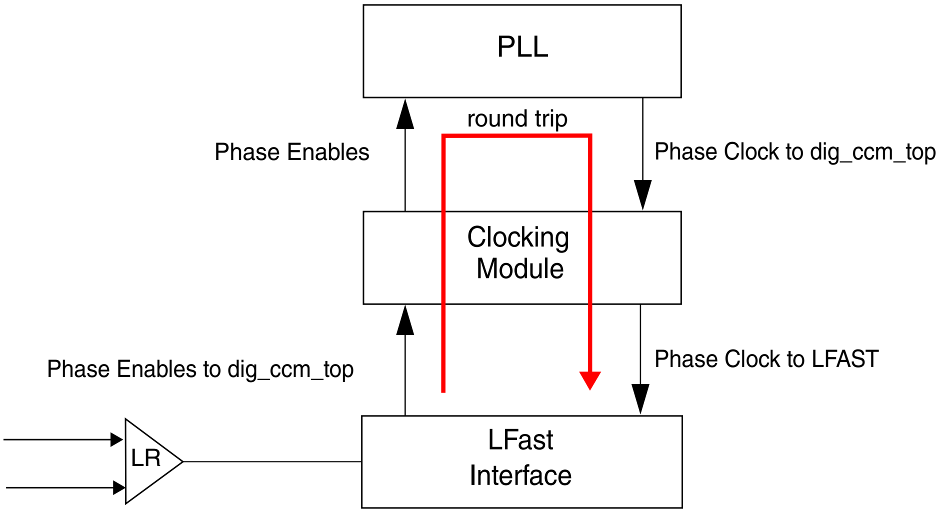


Figure 48-27. Clock enables and clock paths

The LFAST block will know which of the phases will be required for the different modes of operations. Therefore the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

48.10.5 Clock module requirements for low speed phases

48.10.5.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 48-14. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

48.10.6 Tx Controller Clocks

48.10.6.1 Clocking Module Requirements for Speed Phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See the following table.

Table 48-15. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

48.10.6.2 Transmit Clock Muxing

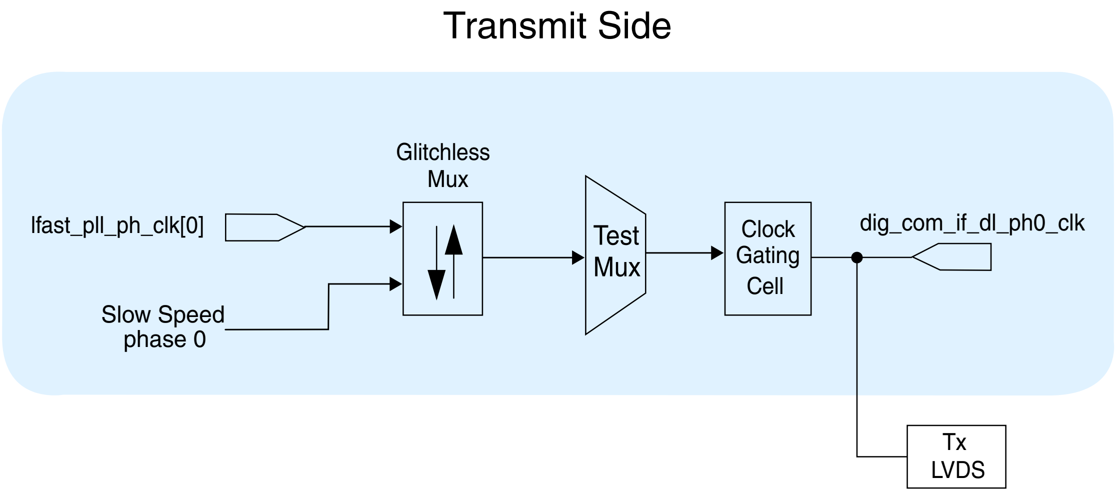


Figure 48-28. Transmit Clocks Muxing

48.10.6.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

Chapter 49
Serial Peripheral Interface (SPI)

49.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The serial peripheral interface (SPI) module provides a synchronous serial bus for communication between a chip and an external peripheral device.

49.1.1 Block Diagram

The block diagram of this module is as follows:

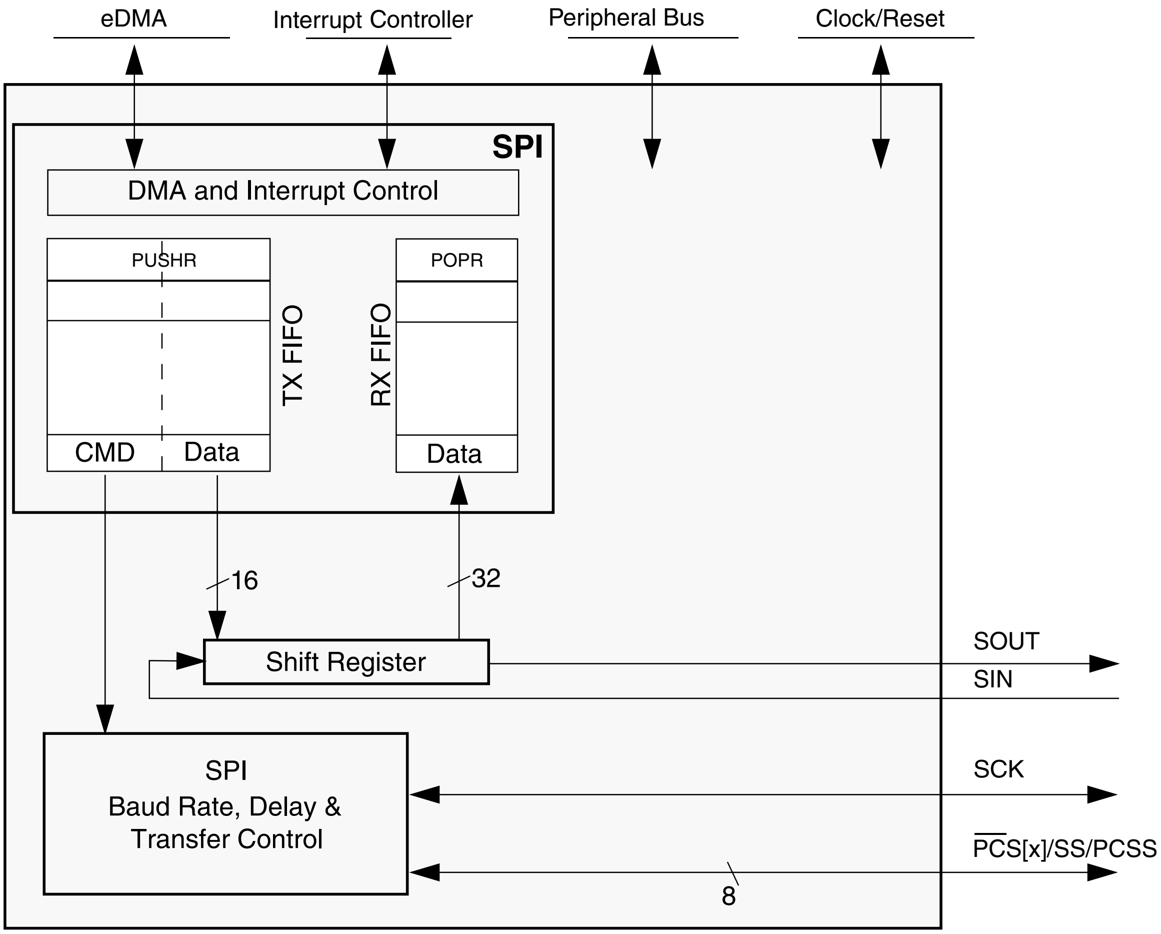


Figure 49-1. SPI Block Diagram

49.1.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Slave mode
- Data streaming operation in Slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 5 entries
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 5 entries
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues

- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
    - four transfer attribute registers
    - Serial clock (SCK) with programmable polarity and phase
    - Various programmable delays
    - Programmable serial frame size: 4 to 16 bits
        - SPI frames longer than 16 bits can be supported using the continuous selection format.
    - Continuously held chip select capability
- 8 peripheral chip selects (PCSes), expandable to 256 with external demultiplexer
- Deglitching support for up to 128 peripheral chip selects (PCSes) with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
    - TX FIFO is not full (TFFF)
    - RX FIFO is not empty (RFDF)
- Interrupt conditions:
    - End of Queue reached (EOQF)
    - TX FIFO is not full (TFFF)
    - Transfer of current frame complete (TCF)
    - Attempt to transmit with an empty Transmit FIFO (TFUF)
    - RX FIFO is not empty (RFDF)
    - Frame received while Receive FIFO is full (RFOF)
- Global interrupt request line
- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features:
    - Support for Stop mode

49.1.3 Interface configurations

49.1.3.1 SPI configuration

The Serial Peripheral Interface (SPI) configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, SPI, and external queues in system RAM.

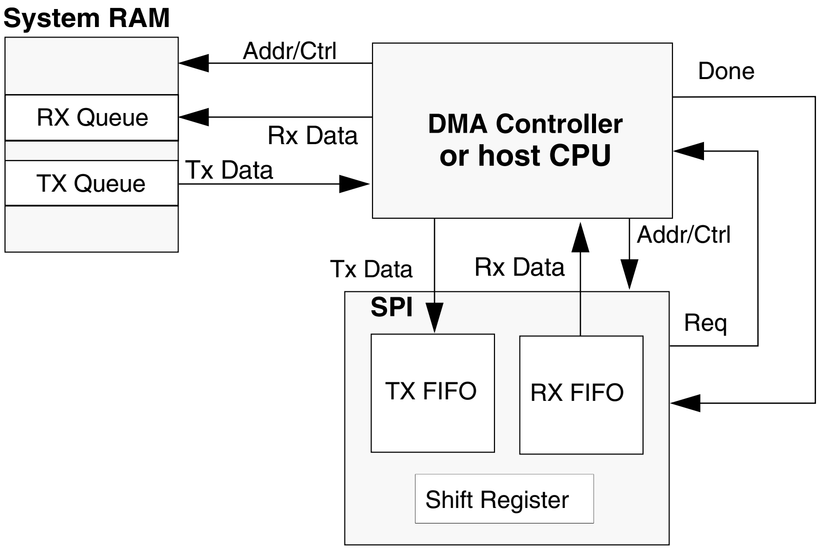


Figure 49-2. SPI with queues and DMA

49.1.4 Modes of Operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:
  - Master mode

    - Slave mode
    - Module Disable mode
- Chip-specific modes:
    - External Stop mode
    - Debug mode

The module enters module-specific modes when the host writes a module register. The chip-specific modes are controlled by signals external to the module. The chip-specific modes are modes that a chip may enter in parallel to the block-specific modes.

49.1.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, these signals are controlled by the module and configured as outputs:

- SCK
- SOUT
- PCS[*x*]

49.1.4.2 Slave Mode

Slave mode allows the module to communicate with SPI bus masters. In this mode, the module responds to externally controlled serial transfers. The SCK signal and the PCS[0]/$\overline{\text{SS}}$ signals are configured as inputs and driven by an SPI bus master.

49.1.4.3 Module Disable Mode

The Module Disable mode can be used for chip power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

49.1.4.4 External Stop Mode

External Stop mode is used for chip power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the protocol clock to the module may be shut off.

49.1.4.5 Debug Mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls module behavior in the Debug mode:

- If the bit is set, the module stops all serial transfers, when the chip is in debug mode.
- If the bit is cleared, the chip debug mode has no effect on the module.

49.2 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

Table 49-1. Module signal descriptions

Signal	Master mode	Slave mode	I/O
PCS0/$\overline{\text{SS}}$	Peripheral Chip Select 0 (O)	Slave Select (I)	I/O
PCS[1:3]	Peripheral Chip Selects 1–3	(Unused)	O
PCS4	Peripheral Chip Select 4	(Unused)	O
PCS5/ $\overline{\text{PCSS}}$	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	(Unused)	O
PCS[6:7]	Peripheral Chip Selects 6–7	(Unused)	O
SCK	Serial Clock (O)	Serial Clock (I)	I/O
SIN	Serial Data In	Serial Data In	I
SOUT	Serial Data Out	Serial Data Out	O

49.2.1 PCS0/$\overline{\text{SS}}$—Peripheral Chip Select/Slave Select

Master mode: Peripheral Chip Select 0 (O)—Selects an SPI slave to receive data transmitted from the module.

Slave mode: Slave Select (I)—Selects the module to receive data transmitted from an SPI master.

49.2.2 PCS1–PCS3—Peripheral Chip Selects 1–3

Master mode: Peripheral Chip Selects 1–3 (O)—Select an SPI slave to receive data transmitted by the module.

Slave mode: Unused

49.2.3 PCS4—Peripheral Chip Select 4

Master mode: Peripheral Chip Select 4 (O)—Selects an SPI slave to receive data transmitted by the module.

Slave mode: Unused

49.2.4 PCS5/$\overline{\text{PCSS}}$—Peripheral Chip Select 5/Peripheral Chip Select Strobe

Master mode:

- Peripheral Chip Select 5 (O)—Used only when the peripheral-chip-select strobe is disabled (MCR[PCSSE]). Selects an SPI slave to receive data transmitted by the module.
- Peripheral Chip Select Strobe (O)—Used only when the peripheral-chip-select strobe is enabled (MCR[PCSSE]). Strobes an off-module peripheral-chip-select demultiplexer, which decodes the module's PCS signals other than PCS5, preventing glitches on the demultiplexer outputs.

Slave mode: Unused

49.2.5 PCS6–PCS7—Peripheral Chip Selects 6–7

Master mode: Peripheral Chip Selects 6–7 (O)—Select an SPI slave to receive data transmitted by the module.

Slave mode: Unused

49.2.6 SCK—Serial Clock

Master mode: Serial Clock (O)—Supplies a clock signal from the module to SPI slaves.

Slave mode: Serial Clock (I)—Supplies a clock signal to the module from an SPI master.

49.2.7 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

Slave mode: Serial Input (I)—Receives serial data.

49.2.8 SOUT—Serial Output

Master mode: Serial Output (O)—Transmits serial data.

Slave mode: Serial Output (O)—Transmits serial data.

NOTE

Serial Data Out output buffers are controlled through SIU (or SIUL) and cannot be controlled through the module.

49.3 Memory Map/Register Definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the POPR and RXFRn also results in a transfer error.

SPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (SPI_MCR)	32	R/W	0000_4001h	49.3.1/1584
8	Transfer Count Register (SPI_TCR)	32	R/W	0000_0000h	49.3.2/1587
C	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR0)	32	R/W	7800_0000h	49.3.3/1587
C	Clock and Transfer Attributes Register (In Slave Mode) (SPI_CTAR0_SLAVE)	32	R/W	7800_0000h	49.3.4/1592
10	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR1)	32	R/W	7800_0000h	49.3.3/1587
14	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR2)	32	R/W	7800_0000h	49.3.3/1587
18	Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR3)	32	R/W	7800_0000h	49.3.3/1587
2C	Status Register (SPI_SR)	32	R/W	0200_0000h	49.3.5/1594
30	DMA/Interrupt Request Select and Enable Register (SPI_RSER)	32	R/W	0000_0000h	49.3.6/1597

Table continues on the next page...

SPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	PUSH TX FIFO Register In Master Mode (SPI_PUSHR)	32	R/W	0000_0000h	49.3.7/1599
34	PUSH TX FIFO Register In Slave Mode (SPI_PUSHR_SLAVE)	32	R/W	0000_0000h	49.3.8/1600
38	POP RX FIFO Register (SPI_POPR)	32	R	0000_0000h	49.3.9/1601
3C	Transmit FIFO Registers (SPI_TXFR0)	32	R	0000_0000h	49.3.10/ 1602
40	Transmit FIFO Registers (SPI_TXFR1)	32	R	0000_0000h	49.3.10/ 1602
44	Transmit FIFO Registers (SPI_TXFR2)	32	R	0000_0000h	49.3.10/ 1602
48	Transmit FIFO Registers (SPI_TXFR3)	32	R	0000_0000h	49.3.10/ 1602
4C	Transmit FIFO Registers (SPI_TXFR4)	32	R	0000_0000h	49.3.10/ 1602
7C	Receive FIFO Registers (SPI_RXFR0)	32	R	0000_0000h	49.3.11/ 1602
80	Receive FIFO Registers (SPI_RXFR1)	32	R	0000_0000h	49.3.11/ 1602
84	Receive FIFO Registers (SPI_RXFR2)	32	R	0000_0000h	49.3.11/ 1602
88	Receive FIFO Registers (SPI_RXFR3)	32	R	0000_0000h	49.3.11/ 1602
8C	Receive FIFO Registers (SPI_RXFR4)	32	R	0000_0000h	49.3.11/ 1602

49.3.1 Module Configuration Register (SPI_MCR)

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSTR	CONT_SCKE	DCONF		FRZ	MTFE	PCSSE	ROOE	PCSIS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	MDIS	DIS_ TXF	DIS_ RXF	0	0	SMPL_PT		0					Reserved	Reserved	HALT
W					CLR_TXF	CLR_RXF										
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

SPI_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Enables either Master mode (if supported) or Slave mode (if supported) operation. 0 Enables Slave mode 1 Enables Master mode
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously.

Table continues on the next page...

SPI_MCR field descriptions (continued)

Field	Description
	0 Continuous SCK disabled. 1 Continuous SCK enabled.
2–3 DCONF	SPI Configuration. Selects among the different configurations of the module. 00 SPI 01 Reserved 10 Reserved 11 Reserved
4 FRZ	Freeze Enables transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in Debug mode. 1 Halt serial transfers in Debug mode.
5 MTFE	Modified Transfer Format Enable Enables a modified transfer format to be used. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS5/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS5/ $\overline{\text{PCSS}}$ is used as the Peripheral Chip Select[5] signal. 1 PCS5/ $\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal.
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8–15 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx when the module is in Master Mode. This field has no effect when the module is in Slave Mode. The Slave Select input to the module in slave mode is always Active Low. **NOTE:** The effect of this bit only takes place when module is enabled. Ensure that this bit is configured correctly before enabling the DSPI interface. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
16 Reserved	This field is reserved.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default

Table continues on the next page...

SPI_MCR field descriptions (continued)

Field	Description
	reset value of 1. When the module is used in Slave Mode, it is recommended to leave this bit 0, because a slave doesn't have control over master transactions. 0 Enables the module clocks. 1 Allows external logic to disable the module clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared. 0 TX FIFO is enabled. 1 TX FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 RX FIFO is enabled. 1 RX FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.
21 CLR_RXF	CLR_RXF Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.
22–23 SMPL_PT	Sample Point Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0. 00 0 protocol clock cycles between SCK edge and SIN sample 01 1 protocol clock cycle between SCK edge and SIN sample 10 2 protocol clock cycles between SCK edge and SIN sample 11 Reserved
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved.
30 Reserved	This field is reserved.
31 HALT	Halt The HALT bit starts and stops frame transfers. See Start and Stop of Module transfers

Table continues on the next page...

SPI_MCR field descriptions (continued)

Field	Description
	0 Start transfers. 1 Stop transfers.

49.3.2 Transfer Count Register (SPI_TCR)

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

Address: 0h base + 8h offset = 8h

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>R</td><td colspan="16" rowspan="2">SPI_TCNT</td><td colspan="16">0</td></tr>
<tr><td>W</td><td colspan="16">▨</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

SPI_TCR field descriptions

Field	Description
0–15 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

49.3.3 Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR*n*)

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 are used to set the slave transfer attributes.

When the module is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used. When the module is configured as an SPI bus slave, it uses the CTAR0 register.

Address: 0h base + Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_CTAR*n* field descriptions

Field	Description
0 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate. **Table 49-2. SPI SCK Duty Cycle** (see below) 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
1–4 FMSZ	Frame Size

Table 49-2. SPI SCK Duty Cycle

DBR	CPHA	PBR	SCK Duty Cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table continues on the next page...

SPI_CTAR*n* field descriptions (continued)

Field	Description
	The number of bits transferred per frame is equal to the FMSZ value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4. There is a constraint on the minimum allowable Frame size, which depends on the ratio of the Register Read/Write clock to the Protocol clock. The minimum Frame size (FMSZ + 1) is determined by the following equation. Upper Ceiling must be applied for non-integer values. Minimum Frame Size = ((4 x f_p) + (3 x f_r)) / (n x f_r) f_p = Protocol clock frequency f_r = Register interface clock frequency n = (PBR x BR) / (1 + DBR) = Multiple of protocol clock required to get Baud (SCK) Clock However, the minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Also note that \`fp' can be equal to, less than or greater than \`fr' purely based on user requirement.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranted. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8–9 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay (t_{CSC}) for more details. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.

Table continues on the next page...

SPI_CTAR*n* field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>10–11
PASC</td><td>After SCK Delay Prescaler

Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. Refer After SCK Delay (t_{ASC}) for more details.

00 Delay after Transfer Prescaler value is 1.
01 Delay after Transfer Prescaler value is 3.
10 Delay after Transfer Prescaler value is 5.
11 Delay after Transfer Prescaler value is 7.</td></tr>
<tr><td>12–13
PDT</td><td>Delay after Transfer Prescaler

Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. See the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer (t_{DT}) for more details.

00 Delay after Transfer Prescaler value is 1.
01 Delay after Transfer Prescaler value is 3.
10 Delay after Transfer Prescaler value is 5.
11 Delay after Transfer Prescaler value is 7.</td></tr>
<tr><td>14–15
PBR</td><td>Baud Rate Prescaler

Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate.

00 Baud Rate Prescaler value is 2.
01 Baud Rate Prescaler value is 3.
10 Baud Rate Prescaler value is 5.
11 Baud Rate Prescaler value is 7.</td></tr>
<tr><td>16–19
CSSCK</td><td>PCS to SCK Delay Scaler

Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation:

$t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$.

The following table lists the delay scaler values.

<b>Table 49-3. Delay Scaler Encoding</b>
<table>
<tr><th>Field Value</th><th>Delay Scaler Value</th></tr>
<tr><td>0000</td><td>2</td></tr>
<tr><td>0001</td><td>4</td></tr>
<tr><td>0010</td><td>8</td></tr>
<tr><td>0011</td><td>16</td></tr>
<tr><td>0100</td><td>32</td></tr>
<tr><td>0101</td><td>64</td></tr>
<tr><td>0110</td><td>128</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

SPI_CTAR*n* field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<b>Table 49-3. Delay Scaler Encoding (continued)</b>
<table>
<tr><th>Field Value</th><th>Delay Scaler Value</th></tr>
<tr><td>0111</td><td>256</td></tr>
<tr><td>1000</td><td>512</td></tr>
<tr><td>1001</td><td>1024</td></tr>
<tr><td>1010</td><td>2048</td></tr>
<tr><td>1011</td><td>4096</td></tr>
<tr><td>1100</td><td>8192</td></tr>
<tr><td>1101</td><td>16384</td></tr>
<tr><td>1110</td><td>32768</td></tr>
<tr><td>1111</td><td>65536</td></tr>
</table>
Refer PCS to SCK Delay (t_{CSC}) for more details.
</td></tr>
<tr><td>20–23
ASC</td><td>
After SCK Delay Scaler

Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation:

$t_{ASC} = (1/f_P) \times PASC \times ASC$

See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.
</td></tr>
<tr><td>24–27
DT</td><td>
Delay After Transfer Scaler

Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame.

In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation:

$t_{DT} = (1/f_P) \times PDT \times DT$

See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values.
</td></tr>
<tr><td>28–31
BR</td><td>
Baud Rate Scaler

Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation:

SCK baud rate = (f_P /PBR) x [(1+DBR)/BR]

The following table lists the baud rate scaler values.

<b>Table 49-4. Baud Rate Scaler</b>
<table>
<tr><th>CTARn[BR]</th><th>Baud Rate Scaler Value</th></tr>
<tr><td>0000</td><td>2</td></tr>
<tr><td>0001</td><td>4</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

SPI_CTAR*n* field descriptions (continued)

Field	Description
	Table 49-4. Baud Rate Scaler (continued)

CTARn[BR]	Baud Rate Scaler Value
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

49.3.4 Clock and Transfer Attributes Register (In Slave Mode) (SPI_CTAR*n*_SLAVE)

When the module is configured as an SPI bus slave, the CTAR0 register is used.

Address: 0h base + Ch offset + (0d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	FMSZ				CPOL	CPHA	0		Reserved	Reserved					
W								▨	▨							
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_CTAR*n*_SLAVE field descriptions

Field	Description
0 Reserved	Always write the reset value to this field. This field is reserved.
1–4 FMSZ	Frame Size The number of bits transfered per frame is equal to the FMSZ field value plus 1. Note that the minimum valid value of frame size is 4.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranted. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	This field is reserved.
10–31 Reserved	This field is reserved.

49.3.5 Status Register (SPI_SR)

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	0	0	0	0	0	RFOF	0	RFDF	0
W	w1c	w1c		w1c	w1c		w1c						w1c		w1c	
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0 Transfer not complete. 1 Transfer complete.
1 TXRXS	TX and RX Status Reflects the run status of the module. 0 Transmit and receive operations are disabled (The module is in Stopped state). 1 Transmit and receive operations are enabled (The module is in Running state).

Table continues on the next page...

SPI_SR field descriptions (continued)

Field	Description
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.
4 TFUF	Transmit FIFO Underflow Flag Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for SPI blocks operating in Slave mode and SPI configuration. TFUF is set when the TX FIFO of the module operating in SPI Slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing 1 to it. 0 No TX FIFO underflow. 1 TX FIFO underflow has occurred.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 TFFF	Transmit FIFO Fill Flag Provides a method for the module to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request. 0 TX FIFO is full. 1 TX FIFO is not full.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SPI_SR field descriptions (continued)

Field	Description
14 RFDF	Receive FIFO Drain Flag Provides a method for the module to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 RX FIFO is empty. 1 RX FIFO is not empty.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–19 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
20–23 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
24–27 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28–31 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

49.3.6 DMA/Interrupt Request Select and Enable Register (SPI_RSER)

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TCF_RE	Reserved	Reserved	EOQF_RE	TFUF_RE	Reserved	TFFF_RE	TFFF_DIRS	Reserved	Reserved	Reserved	Reserved	RFOF_RE	Reserved	RFDF_RE	RFDF_DIRS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	Reserved	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
1 Reserved	Always write the reset value to this field. This field is reserved.
2 Reserved	Always write the reset value to this field. This field is reserved.
3 EOQF_RE	Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request.

Table continues on the next page...

SPI_RSER field descriptions (continued)

Field	Description
	0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
5 Reserved	Always write the reset value to this field. This field is reserved.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
8 Reserved	Always write the reset value to this field. This field is reserved.
9 Reserved	Always write the reset value to this field. This field is reserved.
10 Reserved	Always write the reset value to this field. This field is reserved.
11 Reserved	Always write the reset value to this field. This field is reserved.
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
13 Reserved	Always write the reset value to this field. This field is reserved.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt or DMA requests are disabled. 1 RFDF interrupt or DMA requests are enabled.
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select

Table continues on the next page...

SPI_RSER field descriptions (continued)

Field	Description
	Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
16 Reserved	Always write the reset value to this field. This field is reserved.
17 Reserved	Always write the reset value to this field. This field is reserved.
18–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

49.3.7 PUSH TX FIFO Register In Master Mode (SPI_PUSHR)

Specifies data to be transferred to the TX FIFO. An 8- or 16-bit write access transfers all 32 bits to the TX FIFO. In Master mode, the register transfers 16 bits of data and 16 bits of command information. A read access of PUSHR returns the topmost TX FIFO entry.

When the module is disabled, writing to this register does not update the FIFO. Therefore, any reads performed while the module is disabled return the last PUSHR write performed while the module was still enabled.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CONT	CTAS			EOQ	CTCNT	Reserved		PCS							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1–3 CTAS	Clock and Transfer Attributes Select Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI Slave mode, CTAR0 is used. See the chip specific section for details to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0 001 CTAR1 010 CTAR2 011 CTAR3 100 Reserved 101 Reserved 110 Reserved 111 Reserved
4 EOQ	End Of Queue Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0 Do not clear the TCR[TCNT] field. 1 Clear the TCR[TCNT] field.
6–7 Reserved	Always write the reset value to this field. This field is reserved.
8–15 PCS	PCS Select which PCS signals are to be asserted for the transfer. Refer to the chip-specific SPI information for the number of PCS signals used in this chip. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

49.3.8 PUSH TX FIFO Register In Slave Mode (SPI_PUSHR_SLAVE)

Specifies data to be transferred to the TX FIFO in slave mode. An 8- or 16-bit write access to PUSHR transfers the 16-bit TXDATA field to the TX FIFO.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_PUSHR_SLAVE field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

49.3.9 POP RX FIFO Register (SPI_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_POPR field descriptions

Field	Description
0–31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

49.3.10 Transmit FIFO Registers (SPI_TXFR*n*)

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO.

Address: 0h base + 3Ch offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCMD_TXDATA																TXDATA															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_TXFR*n* field descriptions

Field	Description
0–15 TXCMD_ TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data. In Slave mode, this field is reserved.
16–31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

49.3.11 Receive FIFO Registers (SPI_RXFR*n*)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

NOTE

Accessing the SPI_RXFRn register will cause all subsequent SPI register accesses to be incorrect. Hence, the user should only read received data through, SPI_POPR.

Address: 0h base + 7Ch offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI_RXFR*n* field descriptions

Field	Description
0–31 RXDATA	Receive Data Contains the received SPI data.

49.4 Functional description

The module supports full-duplex, synchronous serial communications between chips and peripheral devices. The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes.

The module has the following configuration

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.

The DCONF field in the Module Configuration Register (MCR) determines the module Configuration. SPI configuration is selected when DCONF within SPIx_MCR is 0b00.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command.

See Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR*n*) for information on the fields of CTAR registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

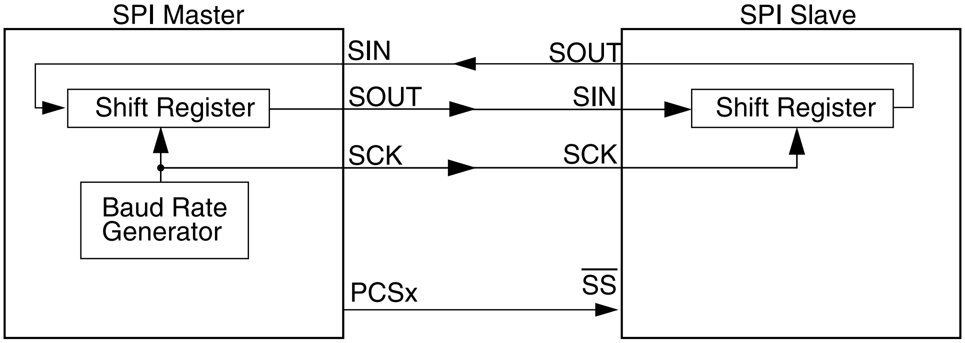


Figure 49-3. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 8 Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with. Refer to the chip specific section for details on the number of PCS signals used in this chip.

The SPI configuration shares transfer protocol and timing properties which are described independently of the configuration in Transfer formats. The transfer rate and delay settings are described in Module baud rate and clock delay generation.

49.4.1 Start and Stop of module transfers

The module has two operating states: Stopped and Running. Both the states are independent of it's configuration. The default state of the module is Stopped. In the Stopped state, no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the module without causing undetermined results. In the Running state serial transfers take place.

The TXRXS bit in the SR indicates the state of module. The bit is set if the module is in Running state.

The module starts or transitions to Running when all of the following conditions are true:

- SR[EOQF] bit is clear
- Chip is not in the Debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The module stops or transitions from Running to Stopped after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- Chip in the Debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

49.4.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to the module RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in the following sections:

- Transmit First In First Out (TX FIFO) buffering mechanism
- Transmit First In First Out (TX FIFO) buffering mechanism
- Receive First In First Out (RX FIFO) buffering mechanism

The interrupt and DMA request conditions are described in Interrupts/DMA requests.

The SPI configuration supports two block-specific modes—Master mode and Slave mode.In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command. In Slave mode, the module responds only to transfers initiated by a bus master external to it and the SPI command field space is reserved.

49.4.2.1 Master mode

In SPI Master mode, the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See PUSH TX FIFO Register In Master Mode (SPI_PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis.

49.4.2.2 Slave mode

In SPI Slave mode the module responds to transfers initiated by an SPI bus master. It does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase, and frame size must be set for successful communication with an SPI master. The SPI Slave mode transfer attributes are set in the CTAR0. The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

49.4.2.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately. Setting the MCR[DIS_TXF] bit disables the TX FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO is disabled:

- SR[TFFF], SR[TFUF] and SR[TXCTR] behave as if there is a one-entry FIFO
- The contents of TXFRs, SR[TXNXTPTR] are undefined

Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

49.4.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 5 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to PUSHR[TXDATA] or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

49.4.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill Interrupt or DMA Request for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

49.4.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a module slave while the slave's TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See Transmit FIFO Underflow Interrupt Request for details.

49.4.2.5 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds 5 received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

49.4.2.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

49.4.2.5.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

49.4.3 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

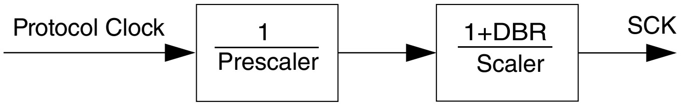


Figure 49-4. Communications clock prescalers and scalers

49.4.3.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 49-5. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

NOTE

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

49.4.3.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 49-6 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

Table 49-6. PCS to SCK delay computation example

f_{SYS}	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

49.4.3.3 After SCK Delay (t_{ASC})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 49-6 and Figure 49-7 for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR*x* registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 49-7. After SCK Delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 μs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

49.4.3.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 49-6 for an illustration of the Delay after Transfer. The PDT and DT fields in the CTAR*x* registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

Table 49-8. Delay after Transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode, the delay is fixed at 1 SCK period.

49.4.3.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The $\overline{\text{PCSS}}$ signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the Module is in Master mode and the PCSSE bit is set in the MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode peripheral chip selects other than PCS5 into glitch-free PCS signals. The following figure shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

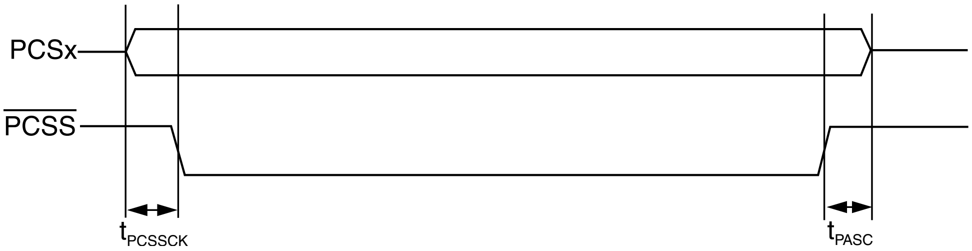


Figure 49-5. Peripheral Chip Select Strobe timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\text{PCSSCK}} = \frac{1}{f_{\text{P}}} \times \text{PCSSCK}$$

At the end of the transfer, the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{\text{PASC}} = \frac{1}{f_{\text{P}}} \times \text{PASC}$$

The following table shows an example of how to compute the t_{pcssck} delay.

Table 49-9. Peripheral Chip Select Strobe Assert computation example

f_P	PCSSCK	Prescaler	Delay before Transfer
100 MHz	0b11	7	70.0 ns

The following table shows an example of how to compute the t_{pasc} delay.

Table 49-10. Peripheral Chip Select Strobe Negate computation example

f_P	PASC	Prescaler	Delay after Transfer
100 MHz	0b11	7	70.0 ns

The $\overline{\text{PCSS}}$ signal is not supported when Continuous Serial Communication SCK mode is enabled.

NOTE

The clock frequency mentioned in the preceding tables is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

49.4.4 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in Slave mode the values of CPOL and CPHA must be identical to the master device settings to ensure proper transmission. In SPI Slave mode, only CTAR0 is used.

The module supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer Format with CPHA = 0
- Modified Transfer Format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The module can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the interface configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

49.4.4.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

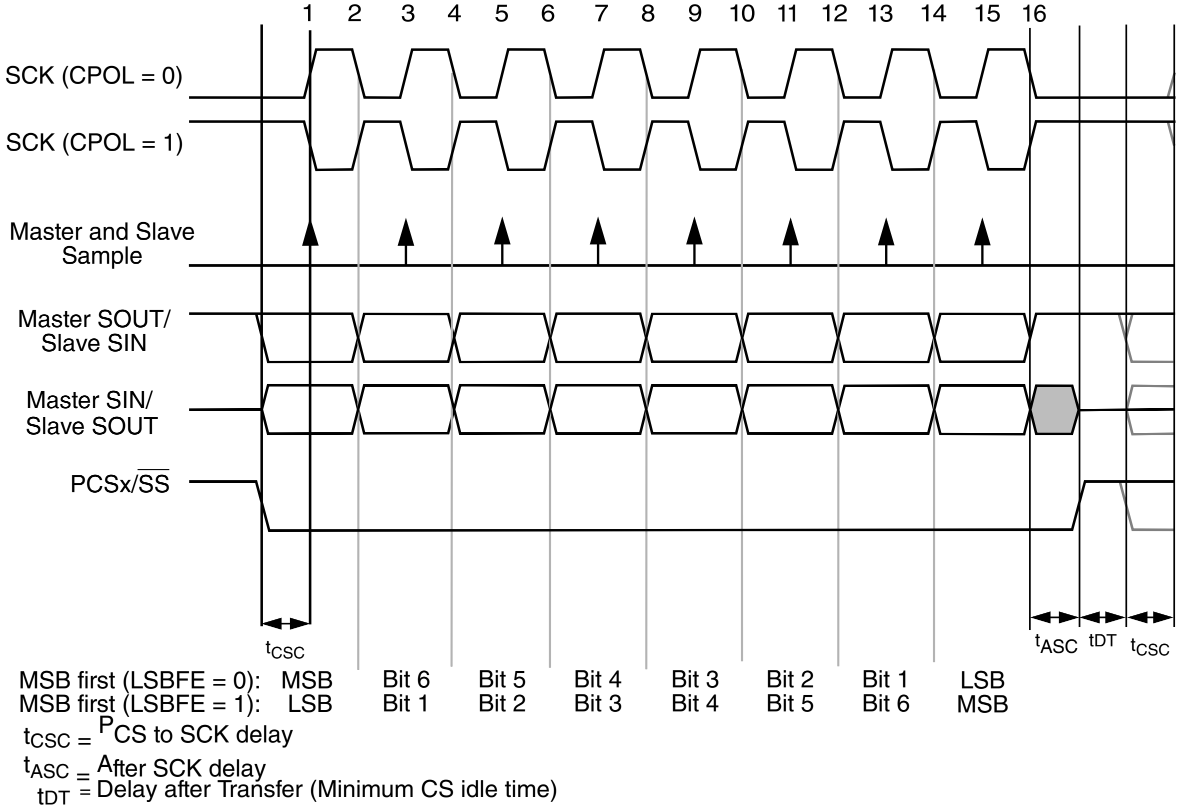

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
SCK (CPOL = 0)
SCK (CPOL = 1)
Master and Slave Sample
Master SOUT/ Slave SIN
Master SIN/ Slave SOUT
PCSx/$\overline{SS}$
t_{CSC}
t_{ASC}
t_{DT}
t_{CSC}
MSB first (LSBFE = 0): MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
MSB first (LSBFE = 1): LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB
t_{CSC} = PCS to SCK delay
t_{ASC} = After SCK delay
t_{DT} = Delay after Transfer (Minimum CS idle time)

Figure 49-6. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

49.4.4.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

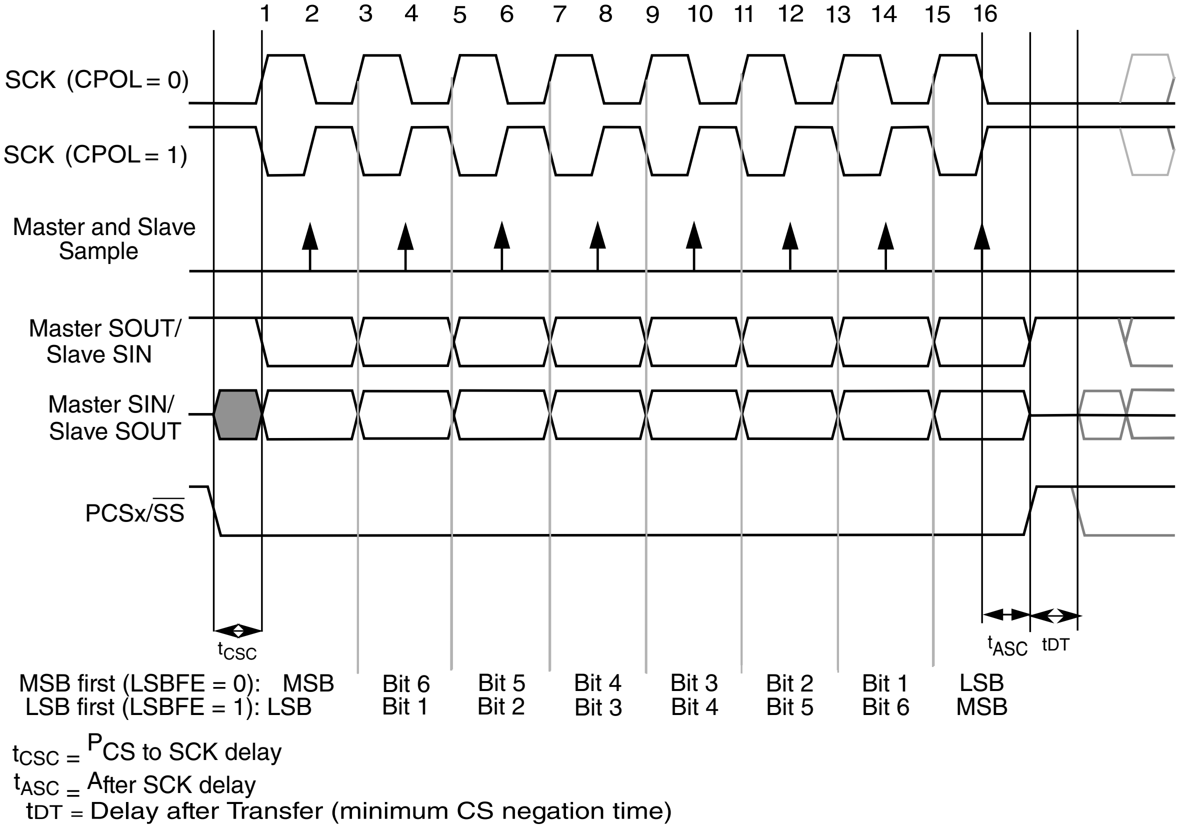


Figure 49-7. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

49.4.4.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actual sample point is delayed by one protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE=1. Timing delays shown are:

- T_{csc} - PCS to SCK assertion delay
- T_{acs} - After SCK PCS negation delay
- T_{su_ms} - master SIN setup time
- T_{hd_ms} - master SIN hold time
- T_{vd_sl} - slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- T_{su_sl} - data setup time on slave data input
- T_{hd_sl} - data hold time on slave data input
- T_{sys} - protocol clock period.

The following figure shows the modified transfer format for CPHA = 0 and Fsys/Fsck = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behavior are shown.

- Signal, marked "SOUT of Ext Slave", presents regular SPI slave serial output.
- Signal, marked "SOUT of DSPI Slave", presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note

In the following diagrams, f_{sys} represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

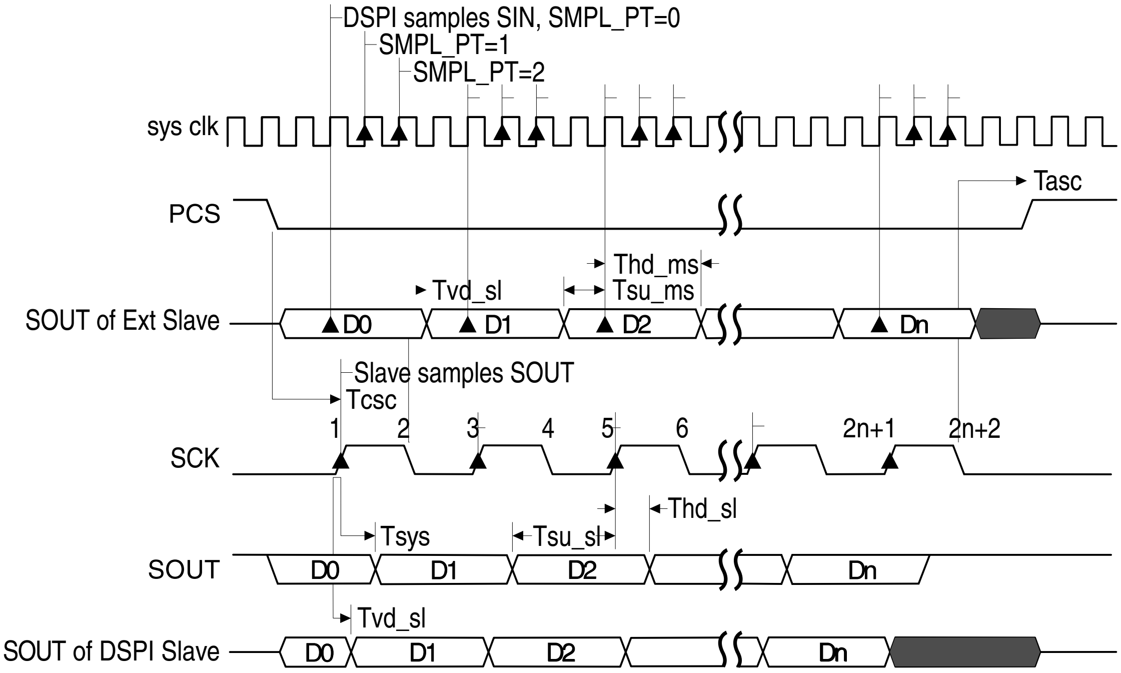


Figure 49-8. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/4$)

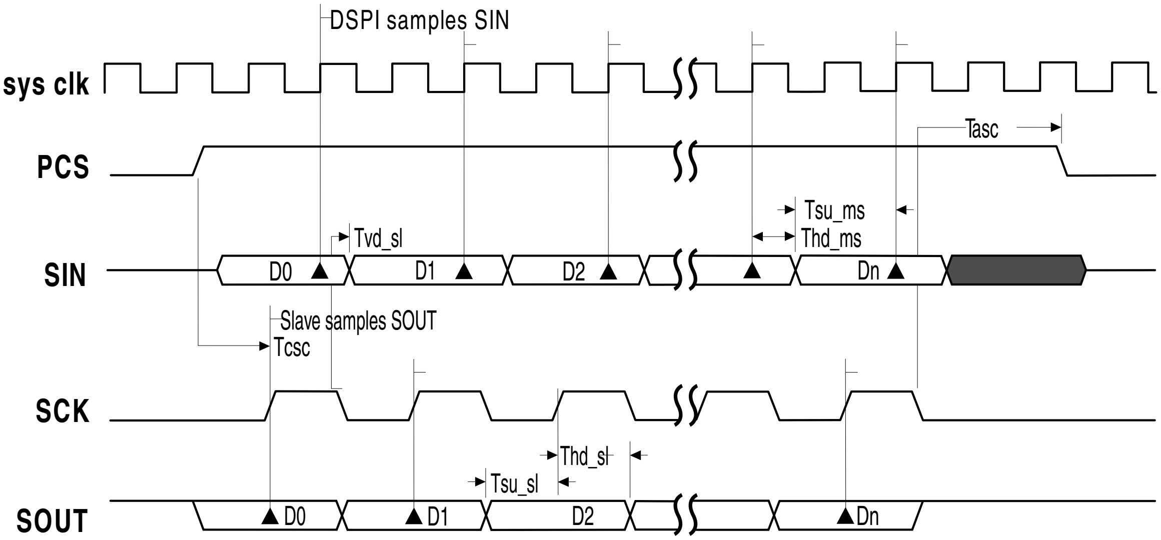


Figure 49-9. DSPI Modified Transfer Format (MTFE=1, CPHA=0, f_{sck} = f_{sys}/2)

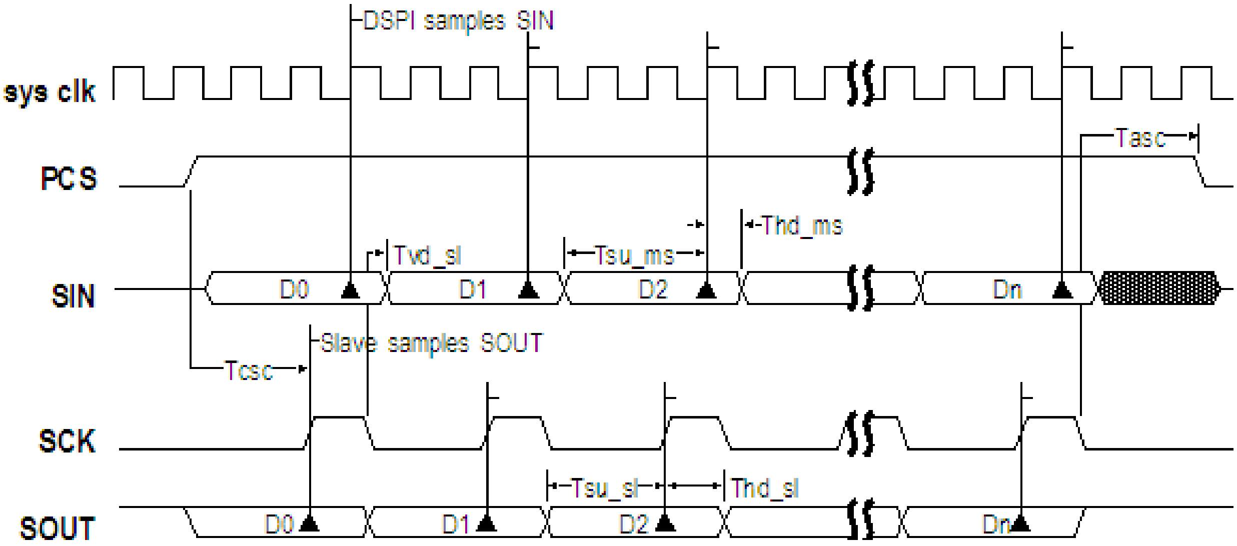


Figure 49-10. DSPI Modified Transfer Format (MTFE=1, CPHA=0, f_{sck} = f_{sys}/3)

49.4.4.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK . The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay and the After SCK delay must be greater or equal to half of the SCK period.**

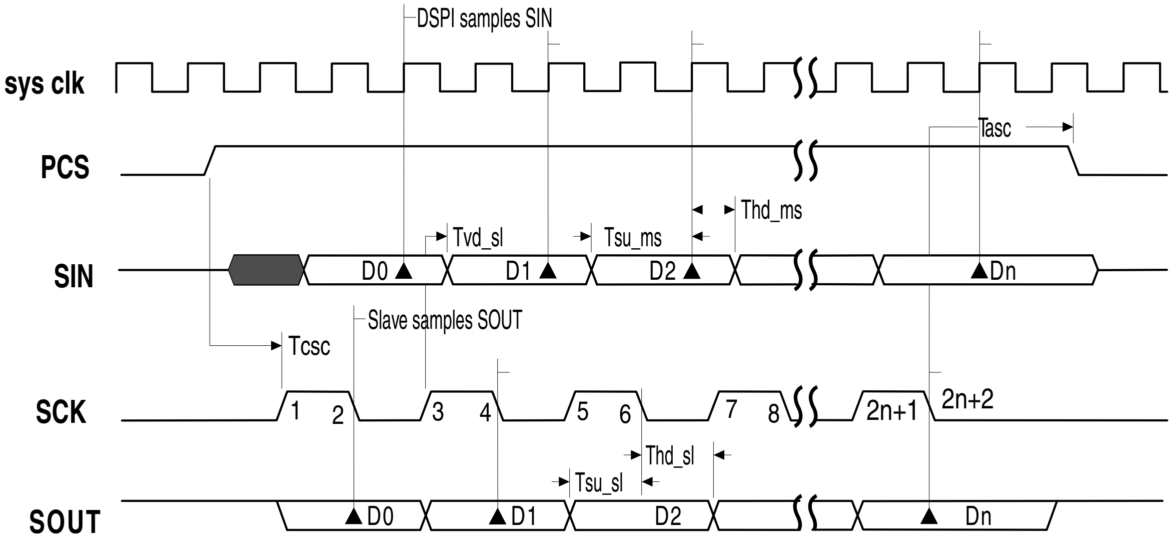


Figure 49-11. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/2$)

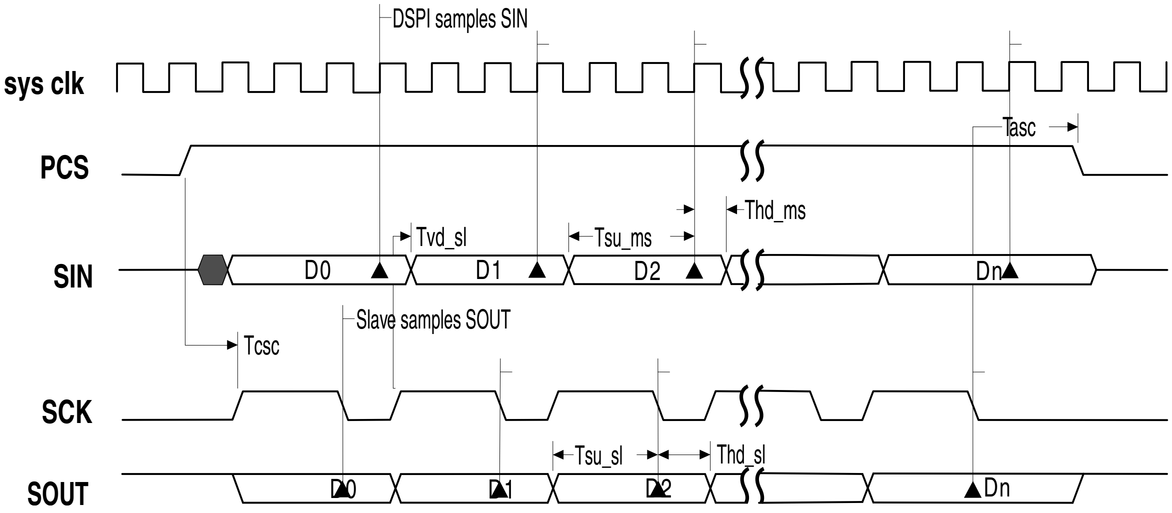


Figure 49-12. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/3$)

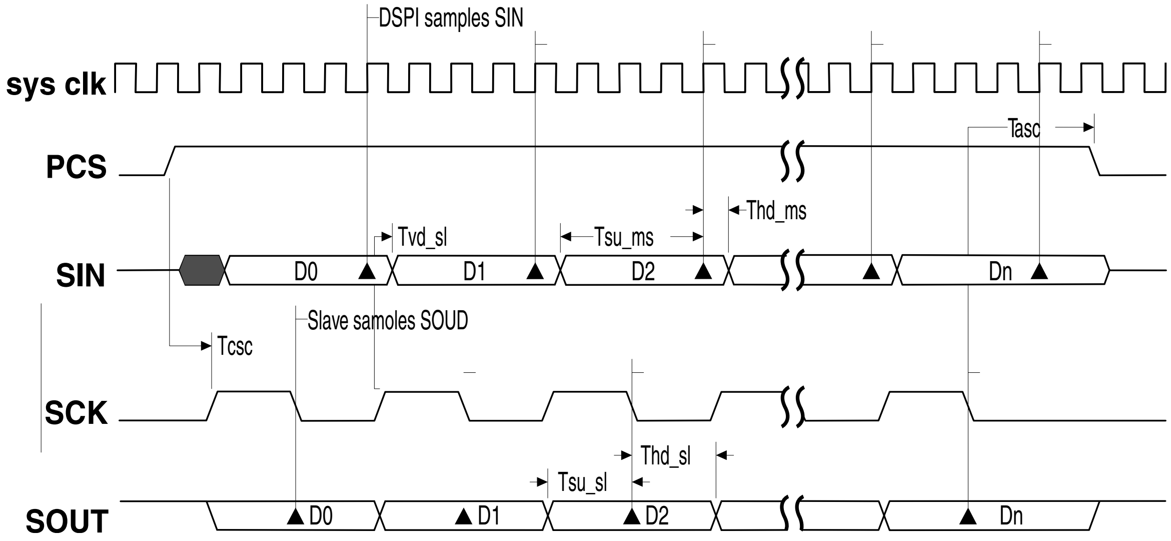


Figure 49-13. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/4$)

49.4.4.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle the following case. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

When the CONT bit = 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The following timing diagram is for two four-bit transfers with CPHA = 1 and CONT = 0.

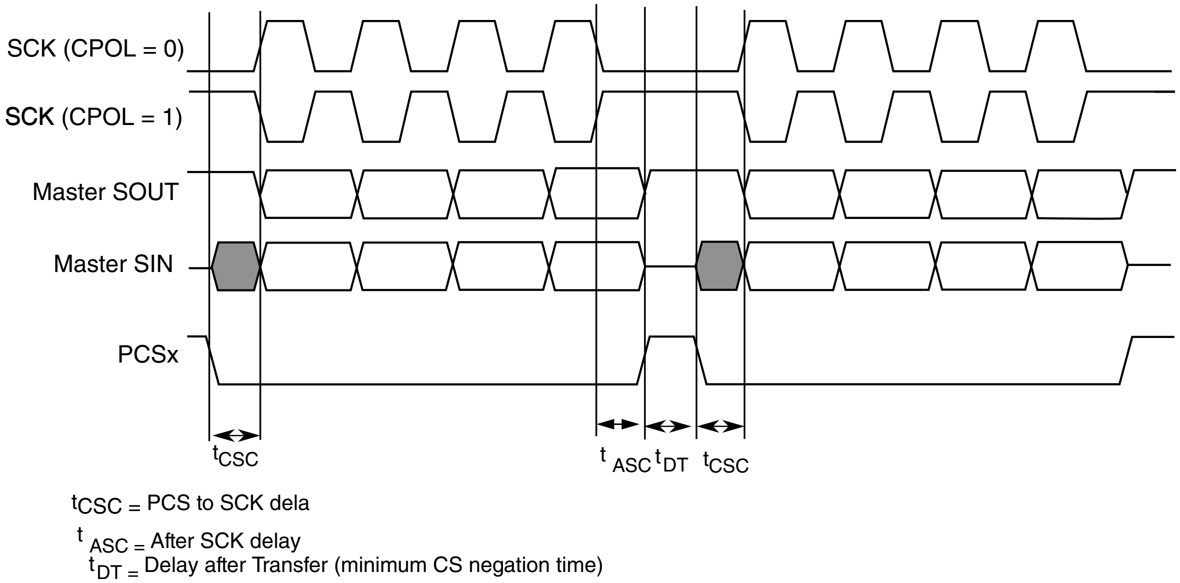


Figure 49-14. Example of non-continuous format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. The following figure shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

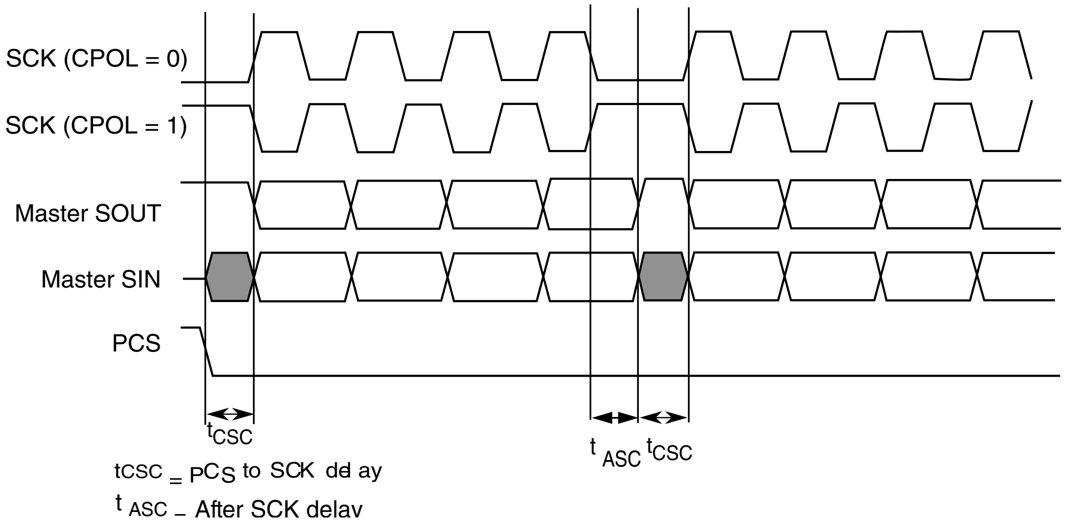


Figure 49-15. Example of continuous transfer (CPHA=1, CONT=1)

When using the module with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.

- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit deasserted in Master mode and the user software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- PUSHR[CONT] must be deasserted before asserting MCR[HALT] in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from Running to Stopped state.

NOTE

User must fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TX FIFO becomes empty.

When operating in Slave mode, ensure that when the last entry in the TX FIFO is completely transmitted, that is, the corresponding TCF flag is asserted and TXFIFO is empty, the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

49.4.5 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

NOTE

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

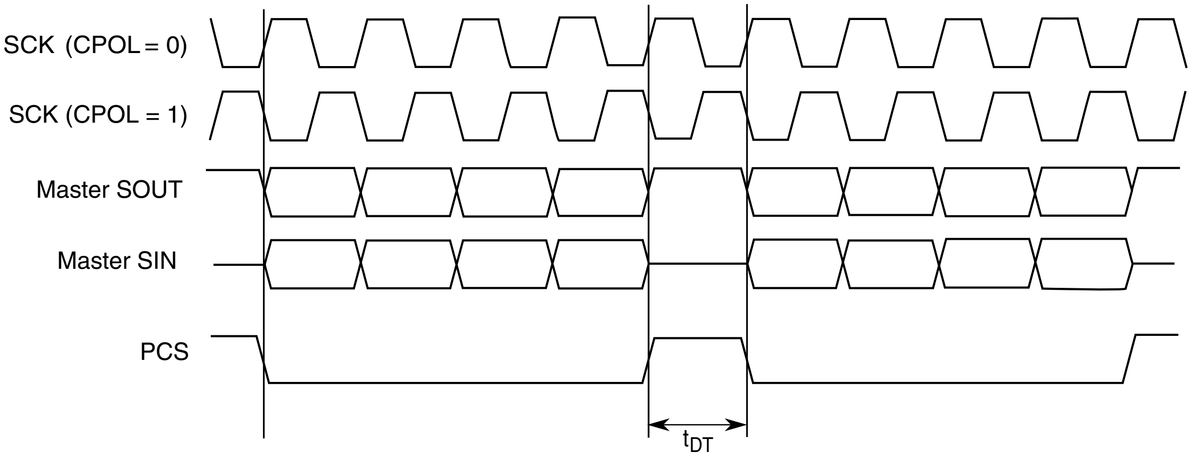


Figure 49-16. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.
- Continuous SCK with CONT bit set and entering Stopped state (refer to Start and Stop of module transfers).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

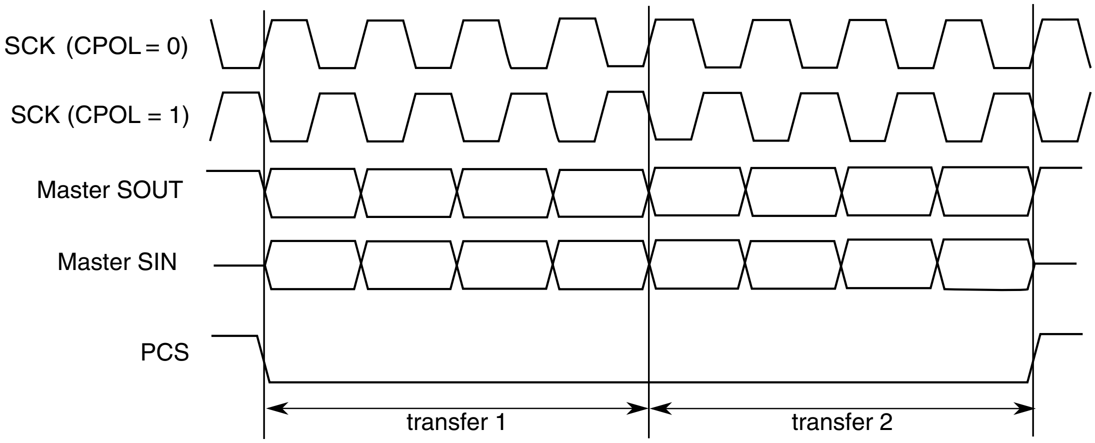


Figure 49-17. Continuous SCK timing diagram (CONT=1)

49.4.6 Slave Mode Operation Constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the module is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{\text{SS}}$ signal is asserted and any time when transmit data is ready and $\overline{\text{SS}}$ signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR register.

If the $\overline{\text{SS}}$ negates before that last SCK edge, the data from shift register is lost.

49.4.7 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

Table 49-11. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	-
TX FIFO Fill	TFFF	Yes	Yes
Transfer Complete	TCF	Yes	-
TX FIFO Underflow	TFUF	Yes	-

Table continues on the next page...

Table 49-11. Interrupt and DMA request conditions (continued)

Condition	Flag	Interrupt	DMA
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	-

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

The module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

49.4.7.1 End Of Queue interrupt request

The End Of Queue (EOQ) interrupt request indicates that the end of a transmit queue is reached. The module generates the interrupt request when EOQ interrupt requests are enabled (RSER[EOQF_RE]) and the EOQ bit in the executing SPI command is 1.

The module generates the interrupt request when the last bit of the SPI frame with EOQ bit set is transmitted.

49.4.7.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

NOTE

> TFFF flag clears automatically when DMA is used to fill TX FIFO.
>
> To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:
>
> 1. Wait until TFFF = 1.
> 2. Write data to PUSHR using CPU.
> 3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

49.4.7.3 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

49.4.7.4 Transmit FIFO Underflow Interrupt Request

The Transmit FIFO Underflow Request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the module operating in Slave mode and SPI configuration . The TFUF bit is set when the TX FIFO of the module is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

49.4.7.5 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

49.4.7.6 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

49.4.8 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

49.4.8.1 Stop mode (External Stop mode)

This module supports the Stop mode protocol. When a request is made to enter External Stop mode, the module acknowledges the request . If a serial transfer is in progress, then this module waits until it reaches the frame boundary before it is ready to have its clocks shut off . While the clocks are shut off, this module's memory-mapped logic is not accessible. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

49.4.8.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware.

When the MDIS bit is set, the module negates the Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state.If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSHR Register does not change the state of the TX FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

49.5 Initialization/application information

This section describes how to initialize the module.

49.5.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO, and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

49.5.2 Switching Master and Slave mode

When changing modes in the module, follow the steps below to guarantee proper operation.

1. Halt it by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a 1 to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable it by clearing MCR[HALT].

49.5.3 Initializing Module in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

49.5.4 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is cleared.

NOTE

> The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 49-12. Baud rate values (bps)

		Baud rate divider prescaler values			
		2	3	5	7
Baud Rate Scaler Values	2	25.0M	16.7M	10.0M	7.14M
	4	12.5M	8.33M	5.00M	3.57M
	6	8.33M	5.56M	3.33M	2.38M
	8	6.25M	4.17M	2.50M	1.79M
	16	3.12M	2.08M	1.25M	893k
	32	1.56M	1.04M	625k	446k
	64	781k	521k	312k	223k
	128	391k	260k	156k	112k
	256	195k	130k	78.1k	55.8k
	512	97.7k	65.1k	39.1k	27.9k
	1024	48.8k	32.6k	19.5k	14.0k
	2048	24.4k	16.3k	9.77k	6.98k

Table continues on the next page...

Table 49-12. Baud rate values (bps) (continued)

		Baud rate divider prescaler values			
		2	3	5	7
	4096	12.2k	8.14k	4.88k	3.49k
	8192	6.10k	4.07k	2.44k	1.74k
	16384	3.05k	2.04k	1.22k	872
	32768	1.53k	1.02k	610	436

49.5.5 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 49-13. Delay values

		Delay prescaler values			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 μs
	32	320.0 ns	960.0 ns	1.6 μs	2.2 μs
	64	640.0 ns	1.9 μs	3.2 μs	4.5 μs
	128	1.3 μs	3.8 μs	6.4 μs	9.0 μs
	256	2.6 μs	7.7 μs	12.8 μs	17.9 μs
	512	5.1 μs	15.4 μs	25.6 μs	35.8 μs
	1024	10.2 μs	30.7 μs	51.2 μs	71.7 μs
	2048	20.5 μs	61.4 μs	102.4 μs	143.4 μs
	4096	41.0 μs	122.9 μs	204.8 μs	286.7 μs
	8192	81.9 μs	245.8 μs	409.6 μs	573.4 μs
	16384	163.8 μs	491.5 μs	819.2 μs	1.1 ms
	32768	327.7 μs	983.0 μs	1.6 ms	2.3 ms
	65536	655.4 μs	2.0 ms	3.3 ms	4.6 ms

49.5.6 Calculation of FIFO pointer addresses

Complete visibility of the FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over. See Transmit First In First Out (TX FIFO) buffering mechanism and Receive First In First Out (RX FIFO) buffering mechanism for details on the FIFO operation.

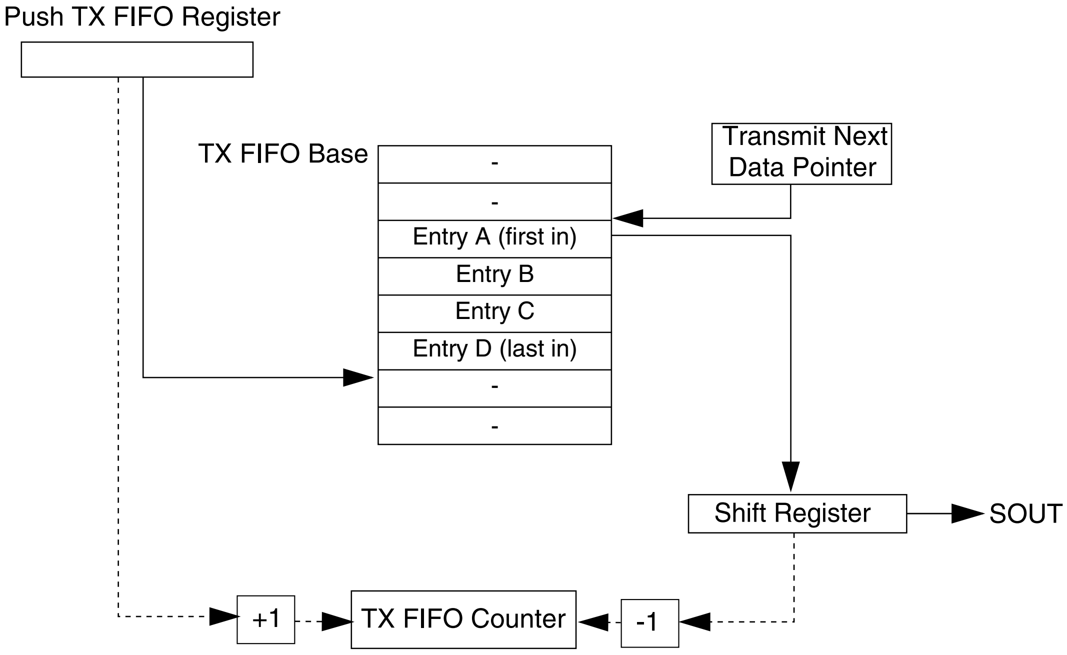


Figure 49-18. TX FIFO pointers and counter

49.5.6.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1)\text{mod}(\text{TXFIFOdepth})$$

TX FIFO Base - Base address of TX FIFO
TXCTR - TX FIFO Counter
TXNXTPTR - Transmit Next Pointer
TX FIFO Depth - Transmit FIFO depth, implementation specific

49.5.6.2 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + (4 \times \text{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1)\text{mod}(\text{RXFIFOdepth})$$

RX FIFO Base - Base address of RX FIFO
RXCTR - RX FIFO counter
POPNXTPTR - Pop Next Pointer
RX FIFO Depth - Receive FIFO depth, implementation specific

Chapter 50
FlexRay Communication Controller (FlexRay)

50.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

This section provides a high-level summary of module features.

50.1.1 Reference

The following documents are referenced.

- *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*[1]
  - Refer to this document for the all of the FlexRay related informaton including the configuration parameters and the allowed parameter ranges.
- *FlexRay Communications System Electrical Physical Layer Specification, Version 3.0*

50.1.2 Glossary

This section provides a list of terms used in this chapter.

[1] The FlexRay Specifications have been developed for automotive applications.The FlexRay Specifications have been neither developed nor tested for non-automotive applications.

Table 50-1. List of terms

Term	Definition
BCU	Buffer Control Unit. Handles message buffer access.
BMIF	Bus Master Interface. Provides master access to FlexRay memory area.
CC	Communication Controller
CDC	Clock Domain Crosser
CHI	Controller Host Interface
Cycle length in µT	The actual length of a cycle in µT for the ideal controller (+/- 0 ppm)
EBI	External Bus Interface
FlexRay Memory Area	Memory area to store the physical message buffer payload data, frame header, frame and slot status, and synchronization frame related tables.
FSS	Frame Start Sequence
HIF	Host Interface. Provides host access to controller.
Host	The FlexRay CC host MCU
Keyslot	Key slot is used to transmit the startup frame, sync frame, or designated single slot frame.
LUT	Look Up Table. Stores message buffer header index value.
LRAM	Look Up Table RAM. Module internal memory to store message buffer configuration data and data field offsets for individual message buffers and receive shadow buffers.
MB	Message Buffer
Mini-slot	An interval of time within the dynamic segment of the communication cycle that is of constant duration (in terms of macroticks) and that is used by the synchronized FTDMA media access scheme to manage media arbitration
MBIDX	Message Buffer Index: the position of a header field entry within the header area. If the header area is accessed as an array, this is the same as the array index of the entry.
MBNum	Message Buffer Number: Position of message buffer configuration registers within the register map. For example, Message Buffer Number 5 corresponds to the MBCCS5 register.
MCU	Microcontroller Unit
µT	Microtick
MT	Macrotick
MTS	Media Access Test Symbol
message frame	Frame with *Null Frame Indicator* set to 1
normal frame	null frame or message frame with both *Sync Frame Indicator* and *Startup Frame Indicator* set to 0
null frame	frame with *Null Frame Indicator* set to 0
NIT	Network Idle Time
PE	Protocol Engine
POC	Protocol Operation Control. Each state of the POC is denoted by POC:state
Rx	Reception
Slot	An interval of time during which access to a communication channel is granted exclusively to a specific node for the transmission of a frame with a frame ID corresponding to the slot.
SEQ	Sequencer Engine
SU	Status update

Table continues on the next page...

Table 50-1. List of terms (continued)

Term	Definition
SECDED	Single-bit error correction, double-bit error detection
System Memory	Memory that contains the FlexRay Memory Area.
System Bus	Bus that connects the controller and System Memory
sync frame	null frame or message frame with *Sync Frame Indicator* set to 1
startup frame	null frame or message frame with both *Sync Frame Indicator* and *Startup Frame Indicator* set to 1
TCU	Time Control Unit
Tx	Transmission

50.1.3 Overview

The CC is a FlexRay communication controller that implements the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

The CC has three main components:

- *Controller host interface (CHI)*
- *Protocol engine (PE)*
- *Clock domain crossing unit (CDC)*

A block diagram of the CC with its surrounding modules is given in the below figure.

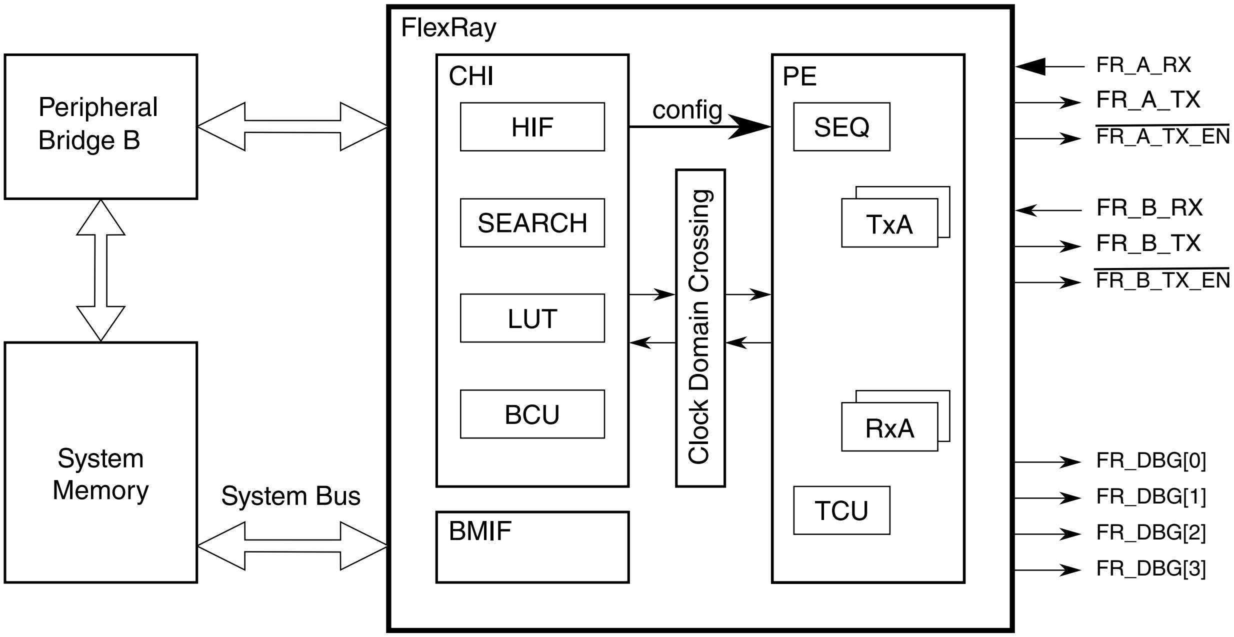


Figure 50-1. FlexRay block diagram

The protocol engine has two transmitter units TxA and TxB and two receiver units RxA and RxB for sending and receiving frames through the two FlexRay channels. The time control unit (TCU) is responsible for maintaining global clock synchronization to the FlexRay network. The overall activity of the PE is controlled by the sequencer engine (SEQ).

The controller host interface provides host access to the module's configuration, control, and status registers, as well as to the message buffer configuration, control, and status registers. The message buffers themselves, which contain the frame header and payload data received or to be transmitted, and the slot status information, are stored in the FlexRay memory area.

The clock domain crossing unit implements signal crossing from the CHI clock domain to the PE clock domain and vice versa, to allow for asynchronous PE and CHI clock domains.

The CC stores the frame header and payload data of frames received or of frames to be transmitted in the FlexRay memory area. The application accesses the FlexRay memory area to retrieve and provide the frames to be processed by the CC. In addition to the frame header and payload data, the CC stores the synchronization frame related tables in the FlexRay memory area for application processing.

The FlexRay memory area is located in the system memory of the MCU. The CC has access to the FlexRay memory area via its bus master interface (BMIF). The host provides the start address of the FlexRay memory area within the system memory by programming the System Memory Base Address Register (FR_SYMBADHR and

FR_SYMBADLR). All FlexRay memory area related offsets are stored in offset registers. The physical address pointer into the FlexRay memory area is calculated using the offset values the FlexRay memory base address.

Note

> The CC does not provide a memory protection scheme for the FlexRay memory area.

50.1.4 Features

The CC provides the following features:

- *FlexRay Communications System Protocol Specification, Version 2.1 Rev A compliant protocol implementation*
- *FlexRay Communications System Electrical Physical Layer Specification, Version 3.0 compliant bus driver interface*
- Single channel support
  - FlexRay Port A can be configured to be connected either to physical FlexRay channel A or physical FlexRay channel B.
- Dual channel support
- FlexRay bus data rates of 10 Mbit/s, 8 Mbit/s, 5 Mbit/s, and 2.5 Mbit/s supported
- 64 configurable message buffers with
  - individual frame ID filtering
  - individual channel ID filtering
  - individual cycle counter filtering
- Message buffer header, status and payload data stored in dedicated FlexRay memory area
  - allows for flexible and efficient message buffer implementation
  - consistent data access ensured by means of buffer locking scheme
  - application can lock multiple buffers at the same time
- Size of message buffer payload data section configurable from 0 up to 254 bytes
- Two independent message buffer segments with configurable size of payload data section

    - each segment can contain message buffers assigned to the static segment and message buffers assigned to the dynamic segment at the same time
- Zero padding for transmit message buffers in static segment
    - applied when the frame payload length exceeds the size of the message buffer data section
- Transmit message buffers configurable with state/event semantics
- Message buffers can be configured as
    - receive message buffer
    - transmit message buffer
- Individual message buffer reconfiguration supported
    - means provided to safely disable individual message buffers
    - disabled message buffers can be reconfigured
- Two independent receive FIFOs
    - one receive FIFO per channel
    - up to 255 entries for each FIFO
    - global frame ID filtering, based on both value/mask filters and range filters
    - global channel ID filtering
    - global message ID filtering for the dynamic segment
- 4 configurable slot error counters
- 4 dedicated slot status indicators
    - used to observe slots without using receive message buffers
- Measured value indicators for the clock synchronization
    - internal synchronization frame ID and synchronization frame measurement tables can be copied into the FlexRay memory area
- Fractional macroticks are supported for clock correction
- Maskable interrupt sources provided via individual and combined interrupt lines
- 1 absolute timer
- 1 timer that can be configured to absolute or relative

- Error correction and error detection (SECDED ECC) for protocol engine data RAM
- Error detection (SECDED ECC) for CHI lookup table RAM

50.1.5 Modes of Operation

This section describes the basic operational power modes of the CC.

50.1.5.1 Disabled Mode

The CC enters the Disabled Mode during hard reset or when the host clears the ‘MEN’ field in the Module Configuration Register (FR_MCR). The host can clear this field by writing ‘0’ and only when the module is in POC:default config mode. The Disabled mode can be checked by reading the module enable field, MEN, in the Module Configuration Register (FR_MCR).

No communication is performed on the FlexRay bus.

All registers with the write access conditions *Any Time* and *Disabled Mode* can be accessed for writing as stated in the Memory Map and Register Description section.

The application configures the CC by accessing the configuration bits and fields in the Module Configuration Register (FR_MCR) as described in Module Initialization .

50.1.5.1.1 Leave Disabled Mode

The CC leaves the Disabled Mode and enters the Normal Mode, when the application writes 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR)

Note

> Once the CC is enabled, it can only be disabled during POC: default config.

50.1.5.2 Normal Mode

In this mode the CC is fully functional. The CC indicates that it is in Normal Mode by asserting the module enable bit MEN in the Module Configuration Register (FR_MCR).

50.1.5.2.1 Enter Normal Mode

This mode is entered when the application requests the CC to leave the Disabled Mode. If the Normal Mode was entered by leaving the Disabled Mode, the application has to perform the protocol initialization described in Protocol Initialization to achieve full FlexRay functionality.

Depending on the values of the SCM, CHA, and CHB bits in the Module Configuration Register (FR_MCR), the corresponding FlexRay bus driver ports are enabled and driven.

50.2 External Signal Description

This section lists and describes the CC signals connected to external pins. These signals are summarized in the following table and described in detail in the Detailed Signal Descriptions section below.

NOTE

Please refer the device configuration of the reference manual for the exact module pin names.

NOTE

The off chip signals FR_A_RX, FR_A_TX, and $\overline{\mathrm{FR\_A\_TX\_EN}}$ are available on each package option. The availability of the other off-chip signals depends on the package option and is chip-specific.

Table 50-2. External Signal Properties

Name	Direction	Active	Reset	Function
FR_A_RX	Input	—	—	Receive Data Channel A
FR_A_TX	Output	—	1	Transmit Data Channel A
$\overline{\mathrm{FR\_A\_TX\_EN}}$	Output	Low	1	Transmit Enable Channel A
FR_B_RX	Input	—	—	Receive Data Channel B
FR_B_TX	Output	—	1	Transmit Data Channel B
$\overline{\mathrm{FR\_B\_TX\_EN}}$	Output	Low	1	Transmit Enable Channel B
FR_DBG[0]	Output	—	0	Debug Strobe Signal 0
FR_DBG[1]	Output	—	0	Debug Strobe Signal 1
FR_DBG[2]	Output	—	0	Debug Strobe Signal 2
FR_DBG[3]	Output	—	0	Debug Strobe Signal 3

50.2.1 Detailed Signal Descriptions

This section provides a detailed description of the CC signals, connected to external pins.

50.2.1.1 FR_A_RX — Receive Data Channel A

The FR_A_RX signal carries the receive data for channel A from the corresponding FlexRay bus driver.

50.2.1.2 FR_A_TX — Transmit Data Channel A

The FR_A_TX signal carries the transmit data for channel A to the corresponding FlexRay bus driver.

50.2.1.3 $\overline{\text{FR\_A\_TX\_EN}}$ — Transmit Enable Channel A

The $\overline{\text{FR\_A\_TX\_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel A.

50.2.1.4 FR_B_RX — Receive Data Channel B

The FR_B_RX signal carries the receive data for channel B from the corresponding FlexRay bus driver.

50.2.1.5 FR_B_TX — Transmit Data Channel B

The FR_B_TX signal carries the transmit data for channel B to the corresponding FlexRay bus driver

50.2.1.6 $\overline{\text{FR\_B\_TX\_EN}}$ — Transmit Enable Channel B

The $\overline{\text{FR\_B\_TX\_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel B.

50.2.1.7 FR_DBG[3], FR_DBG[2], FR_DBG[1], FR_DBG[0] — Strobe Signals

These signals provide the selected debug strobe signals. For details on the debug strobe signal selection refer to the Strobe Signal Support.

50.3 Controller Host Interface Clocking

The clock for the CHI is derived from the system bus clock and has the same phase and frequency as the system bus clock. There are two constraints for the minimum CHI clock frequency.

The first constraint corresponds to the number of utilized message buffers and is specified in Number of Usable Message Buffers.

The second constraint corresponds to the value of the TIMEOUT field in the System Memory Access Time-Out Register (FR_SYMATOR) and is specified in Configure System Memory Access Time-Out Register (FR_SYMATOR).

50.4 Protocol Engine Clocking

The clock for the protocol engine can be generated by two sources. The first source is the internal crystal oscillator and the second source is an internal PLL. The clock source to be used is selected by the clock source select bit CLKSEL in the Module Configuration Register (FR_MCR).

Note

See the Device clocking details for the exact clock sources used.

50.4.1 Oscillator Clocking

If the protocol engine is clocked by the internal crystal oscillator, a crystal or CMOS compatible clock must be connected to the oscillator pins. The crystal or clock must fulfill the requirements given by the *FlexRay Communications System Protocol Specification, Version* 2.1 Rev A.

50.4.2 PLL Clocking

If the protocol engine is clocked by the internal PLL, the PLL clock should be divided down appropriately and provided to the PE.

For more details, see the clocking chapter of the device reference manual.

50.5 Register Descriptions

This section provides detailed descriptions of all registers in ascending address order, presented as 16-bit wide entities.

Table 50-3. Register Access Conventions

Convention	Description
▨	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
R*	Reserved bit or field, will not be changed. Application must not write any value different from the reset value.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.

Table 50-4. Register Field Types

Convention	Description
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A flag bit that can be read, is cleared by writing a one, writing 0 has no effect.
Reset Value	
0	Resets to zero.
1	Resets to one.
-	Not defined after reset and not affected by reset.

50.5.1 Register Reset

All registers except the Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*), Message Buffer Frame ID Register (FR_MBFIDR*n*), and Message Buffer Index Register (FR_MBIDXR*n*) are reset to their reset value on system reset. The registers mentioned above are located in physical memory blocks and, thus, they are not affected by reset. For some register fields, additional reset conditions exist. These additional reset conditions are mentioned in the detailed description of the register. The additional reset conditions are explained in the table below.

Table 50-5. Additional Register Reset Conditions

Condition	Description
Protocol RUN Command	The register field is reset when the application writes to RUN command "0101" to the POCCMD field in the Protocol Operation Control Register (FR_POCR).
Message Buffer Disable	The register field is reset when the application has disabled the message buffer. This happens when the application writes 1 to the message buffer disable trigger bit FR_MBCCSRn[EDT] while the message buffer is enabled (FR_MBCCSRn[EDS] = 1) and the CC grants the disable to the application by clearing the FR_MBCCSRn[EDS] bit.

50.5.2 Register Write Access

This section describes the write access restriction terms that apply to all registers.

50.5.2.1 Register Write Access Restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in the table below. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed. The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled. The condition term [A and B] indicates that the register or field can be written to if both conditions are fulfilled.

Table 50-6. Register Write Access Restrictions

Condition	Indication	Description
Any Time	-	No write access restriction.
Disabled Mode	FR_MCR[MEN] = 0	Write access only when CC is in Disabled Mode.
Normal Mode	FR_MCR[MEN] = 1	Write access only when CC is in Normal Mode.
POC:config	FR_PSR0[PROTSTATE] = *POC:config*	Write access only when Protocol is in the *POC:config* state.
MB_DIS	FR_MBCCSR[EDS] = 0	Write access only when related Message Buffer is disabled.
MB_LCK	FR_MBCCSRn[LCKS] = 1	Write access only when related Message Buffer is locked.
IDL	FR_EEIRICR[BSY] = 0	Write access only when ECC configuration is idle.

50.5.2.2 Register Write Access Requirements

All registers can be accessed with 8-bit, 16-bit and 32-bit wide operations.

For some of the registers, at least a 16-bit wide write access is required to ensure correct operation. This write access requirement is described in the Memory map and register definition section. If an 8-bit wide write access is performed to any of these registers, this access is ignored without notification.

50.5.2.3 Internal Register Access

The following memory mapped registers are used to access multiple internal registers.

- *Strobe Signal Control Register (FR_STBSCR)*
- *Slot Status Selection Register (FR_SSSR)*
- *Slot Status Counter Condition Register (FR_SSCCR)*
- *Receive Shadow Buffer Configuration Data*

Each of these memory mapped registers provides a SEL field and a WMD bit. The SEL field is used to select the internal register. The WMD bit controls the write mode. If the WMD bit is set to 0 during the write access, all fields of the internal register are updated. If the WMD bit set to 1, only the SEL field is changed. All other fields of the internal register remain unchanged. This allows for reading back the values of the selected internal register in a subsequent read access.

50.6 Memory map and register definition

The CC occupies 8 KB (8192 bytes) of address space starting at the CC base address defined by the memory map of the MCU. Address offset - 0x0Ah,0xDEh-0xE0h should not be accessed by application as corresponding feature/s are not available. Therefore, transfer error will not be generated at these offset/s.

NOTE

The following registers are 16-bit write accessible only.

Strobe Signal Control Register (FR_STBSCR)

PE DRAM Access Register (FR_PEDRAR)

PE DRAM Data Register (FR_PEDRDR)

Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Slot Status Selection Register (FR_SSSR)

Slot Status Counter Condition Register (FR_SSCCR)

Receive Shadow Buffer Index Register (FR_RSBIR)

Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Message Buffer Cycle Counter Filter Register (FR_MBCCFRn)

Message Buffer Frame ID Register (FR_MBFIDRn)

Message Buffer Index Register (FR_MBIDXRn)

Message Buffer Data Field Offset Register (FR_MBDORn)

LRAM ECC Error Test Register (FR_LEETRn)

FR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Version Register (FR_MVR)	16	R	A468h	50.6.1/1667
2	Module Configuration Register (FR_MCR)	16	R/W	0000h	50.6.2/1668
4	System Memory Base Address High Register (FR_SYMBADHR)	16	R/W	0000h	50.6.3/1670
6	System Memory Base Address Low Register (FR_SYMBADLR)	16	R/W	0000h	50.6.4/1671
8	Strobe Signal Control Register (FR_STBSCR)	16	R/W	0000h	50.6.5/1671
C	Message Buffer Data Size Register (FR_MBDSR)	16	R/W	0000h	50.6.6/1673
E	Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)	16	R/W	3F3Fh	50.6.7/1674
10	PE DRAM Access Register (FR_PEDRAR)	16	R/W	0000h	50.6.8/1675
12	PE DRAM Data Register (FR_PEDRDR)	16	R/W	0000h	50.6.9/1676
14	Protocol Operation Control Register (FR_POCR)	16	R/W	0000h	50.6.10/ 1676
16	Global Interrupt Flag and Enable Register (FR_GIFER)	16	R/W	0000h	50.6.11/ 1679
18	Protocol Interrupt Flag Register 0 (FR_PIFR0)	16	R/W	0000h	50.6.12/ 1681
1A	Protocol Interrupt Flag Register 1 (FR_PIFR1)	16	R/W	0000h	50.6.13/ 1684

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C	Protocol Interrupt Enable Register 0 (FR_PIER0)	16	R/W	0000h	50.6.14/ 1685
1E	Protocol Interrupt Enable Register 1 (FR_PIER1)	16	R/W	0000h	50.6.15/ 1687
20	CHI Error Flag Register (FR_CHIERFR)	16	R/W	0000h	50.6.16/ 1689
22	Message Buffer Interrupt Vector Register (FR_MBIVEC)	16	R	0000h	50.6.17/ 1691
24	Channel A Status Error Counter Register (FR_CASERCR)	16	R	0000h	50.6.18/ 1692
26	Channel B Status Error Counter Register (FR_CBSERCR)	16	R	0000h	50.6.19/ 1693
28	Protocol Status Register 0 (FR_PSR0)	16	R	0000h	50.6.20/ 1693
2A	Protocol Status Register 1 (FR_PSR1)	16	R	0000h	50.6.21/ 1695
2C	Protocol Status Register 2 (FR_PSR2)	16	R	0000h	50.6.22/ 1696
2E	Protocol Status Register 3 (FR_PSR3)	16	R/W	0000h	50.6.23/ 1699
30	Macrotick Counter Register (FR_MTCTR)	16	R	0000h	50.6.24/ 1701
32	Cycle Counter Register (FR_CYCTR)	16	R	0000h	50.6.25/ 1701
34	Slot Counter Channel A Register (FR_SLTCTAR)	16	R	0000h	50.6.26/ 1702
36	Slot Counter Channel B Register (FR_SLTCTBR)	16	R	0000h	50.6.27/ 1702
38	Rate Correction Value Register (FR_RTCORVR)	16	R	0000h	50.6.28/ 1703
3A	Offset Correction Value Register (FR_OFCORVR)	16	R	0000h	50.6.29/ 1703
3C	Combined Interrupt Flag Register (FR_CIFR)	16	R	0000h	50.6.30/ 1704
3E	System Memory Access Time-Out Register (FR_SYMATOR)	16	R/W	0006h	50.6.31/ 1705
40	Sync Frame Counter Register (FR_SFCNTR)	16	R	0000h	50.6.32/ 1706
42	Sync Frame Table Offset Register (FR_SFTOR)	16	R/W	0000h	50.6.33/ 1707
44	Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)	16	R/W	0000h	50.6.34/ 1707
46	Sync Frame ID Rejection Filter Register (FR_SFIDRFR)	16	R/W	0000h	50.6.35/ 1709

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
48	Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)	16	R/W	0000h	50.6.36/ 1710
4A	Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)	16	R/W	0000h	50.6.37/ 1710
4C	Network Management Vector Register (FR_NMVR0)	16	R	0000h	50.6.38/ 1711
4E	Network Management Vector Register (FR_NMVR1)	16	R	0000h	50.6.38/ 1711
50	Network Management Vector Register (FR_NMVR2)	16	R	0000h	50.6.38/ 1711
52	Network Management Vector Register (FR_NMVR3)	16	R	0000h	50.6.38/ 1711
54	Network Management Vector Register (FR_NMVR4)	16	R	0000h	50.6.38/ 1711
56	Network Management Vector Register (FR_NMVR5)	16	R	0000h	50.6.38/ 1711
58	Network Management Vector Length Register (FR_NMVLR)	16	R/W	0000h	50.6.39/ 1712
5A	Timer Configuration and Control Register (FR_TICCR)	16	R/W	0000h	50.6.40/ 1712
5C	Timer 1 Cycle Set Register (FR_TI1CYSR)	16	R/W	0000h	50.6.41/ 1713
5E	Timer 1 Macrotick Offset Register (FR_TI1MTOR)	16	R/W	0000h	50.6.42/ 1714
60	Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)	16	R/W	0000h	50.6.43/ 1715
60	Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL)	16	R/W	0000h	50.6.44/ 1715
62	Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)	16	R/W	0000h	50.6.45/ 1716
62	Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL)	16	R/W	0000h	50.6.46/ 1717
64	Slot Status Selection Register (FR_SSSR)	16	R/W	0000h	50.6.47/ 1717
66	Slot Status Counter Condition Register (FR_SSCCR)	16	R/W	0000h	50.6.48/ 1719
68	Slot Status Register (FR_SSR0)	16	R	0000h	50.6.49/ 1721
6A	Slot Status Register (FR_SSR1)	16	R	0000h	50.6.49/ 1721
6C	Slot Status Register (FR_SSR2)	16	R	0000h	50.6.49/ 1721
6E	Slot Status Register (FR_SSR3)	16	R	0000h	50.6.49/ 1721

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
70	Slot Status Register (FR_SSR4)	16	R	0000h	50.6.49/ 1721
72	Slot Status Register (FR_SSR5)	16	R	0000h	50.6.49/ 1721
74	Slot Status Register (FR_SSR6)	16	R	0000h	50.6.49/ 1721
76	Slot Status Register (FR_SSR7)	16	R	0000h	50.6.49/ 1721
78	Slot Status Counter Register (FR_SSCR0)	16	R	0000h	50.6.50/ 1722
7A	Slot Status Counter Register (FR_SSCR1)	16	R	0000h	50.6.50/ 1722
7C	Slot Status Counter Register (FR_SSCR2)	16	R	0000h	50.6.50/ 1722
7E	Slot Status Counter Register (FR_SSCR3)	16	R	0000h	50.6.50/ 1722
80	MTS A Configuration Register (FR_MTSACFR)	16	R/W	0000h	50.6.51/ 1723
82	MTS B Configuration Register (FR_MTSBCFR)	16	R/W	0000h	50.6.52/ 1724
84	Receive Shadow Buffer Index Register (FR_RSBIR)	16	R/W	0000h	50.6.53/ 1725
86	Receive FIFO Watermark and Selection Register (FR_RFWMSR)	16	R/W	0000h	50.6.54/ 1726
88	Receive FIFO Start Index Register (FR_RFSIR)	16	R/W	0000h	50.6.55/ 1727
8A	Receive FIFO Depth and Size Register (FR_RFDSR)	16	R/W	0000h	50.6.56/ 1727
8C	Receive FIFO A Read Index Register (FR_RFARIR)	16	R	0000h	50.6.57/ 1728
8E	Receive FIFO B Read Index Register (FR_RFBRIR)	16	R	0000h	50.6.58/ 1728
90	Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)	16	R/W	0000h	50.6.59/ 1729
92	Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)	16	R/W	0000h	50.6.60/ 1730
94	Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)	16	R/W	0000h	50.6.61/ 1730
96	Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)	16	R/W	0000h	50.6.62/ 1731
98	Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)	16	R/W	0000h	50.6.63/ 1731
9A	Receive FIFO Range Filter Control Register (FR_RFRFCTR)	16	R/W	0000h	50.6.64/ 1732

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C	Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)	16	R	0000h	50.6.65/ 1733
9E	Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)	16	R	0000h	50.6.66/ 1734
A0	Protocol Configuration Register 0 (FR_PCR0)	16	R/W	0000h	50.6.67/ 1734
A2	Protocol Configuration Register 1 (FR_PCR1)	16	R/W	0000h	50.6.68/ 1737
A4	Protocol Configuration Register 2 (FR_PCR2)	16	R/W	0000h	50.6.69/ 1737
A6	Protocol Configuration Register 3 (FR_PCR3)	16	R/W	0000h	50.6.70/ 1738
A8	Protocol Configuration Register 4 (FR_PCR4)	16	R/W	0000h	50.6.71/ 1739
AA	Protocol Configuration Register 5 (FR_PCR5)	16	R/W	0000h	50.6.72/ 1739
AC	Protocol Configuration Register 6 (FR_PCR6)	16	R/W	0000h	50.6.73/ 1740
AE	Protocol Configuration Register 7 (FR_PCR7)	16	R/W	0000h	50.6.74/ 1741
B0	Protocol Configuration Register 8 (FR_PCR8)	16	R/W	0000h	50.6.75/ 1741
B2	Protocol Configuration Register 9 (FR_PCR9)	16	R/W	0000h	50.6.76/ 1742
B4	Protocol Configuration Register 10 (FR_PCR10)	16	R/W	0000h	50.6.77/ 1743
B6	Protocol Configuration Register 11 (FR_PCR11)	16	R/W	0000h	50.6.78/ 1744
B8	Protocol Configuration Register 12 (FR_PCR12)	16	R/W	0000h	50.6.79/ 1744
BA	Protocol Configuration Register 13 (FR_PCR13)	16	R/W	0000h	50.6.80/ 1745
BC	Protocol Configuration Register 14 (FR_PCR14)	16	R/W	0000h	50.6.81/ 1746
BE	Protocol Configuration Register 15 (FR_PCR15)	16	R/W	0000h	50.6.82/ 1746
C0	Protocol Configuration Register 16 (FR_PCR16)	16	R/W	0000h	50.6.83/ 1747
C2	Protocol Configuration Register 17 (FR_PCR17)	16	R/W	0000h	50.6.84/ 1747
C4	Protocol Configuration Register 18 (FR_PCR18)	16	R/W	0000h	50.6.85/ 1748
C6	Protocol Configuration Register 19 (FR_PCR19)	16	R/W	0000h	50.6.86/ 1748

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C8	Protocol Configuration Register 20 (FR_PCR20)	16	R/W	0000h	50.6.87/ 1749
CA	Protocol Configuration Register 21 (FR_PCR21)	16	R/W	0000h	50.6.88/ 1749
CC	Protocol Configuration Register 22 (FR_PCR22)	16	R/W	0000h	50.6.89/ 1750
CE	Protocol Configuration Register 23 (FR_PCR23)	16	R/W	0000h	50.6.90/ 1750
D0	Protocol Configuration Register 24 (FR_PCR24)	16	R/W	0000h	50.6.91/ 1751
D2	Protocol Configuration Register 25 (FR_PCR25)	16	R/W	0000h	50.6.92/ 1751
D4	Protocol Configuration Register 26 (FR_PCR26)	16	R/W	0000h	50.6.93/ 1752
D6	Protocol Configuration Register 27 (FR_PCR27)	16	R/W	0000h	50.6.94/ 1753
D8	Protocol Configuration Register 28 (FR_PCR28)	16	R/W	0000h	50.6.95/ 1753
DA	Protocol Configuration Register 29 (FR_PCR29)	16	R/W	0000h	50.6.96/ 1754
DC	Protocol Configuration Register 30 (FR_PCR30)	16	R/W	0000h	50.6.97/ 1754
E2	Protocol Event Output Enable Register (FR_PEOER)	16	R/W	0000h	50.6.98/ 1755
E6	Receive FIFO Start Data Offset Register (FR_RFSDOR)	16	R/W	0000h	50.6.99/ 1756
E8	Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)	16	R/W	0000h	50.6.100/ 1756
EA	Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)	16	R/W	0000h	50.6.101/ 1757
EC	Receive FIFO Periodic Timer Register (FR_RFPTR)	16	R/W	0000h	50.6.102/ 1757
EE	Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)	16	R/W	0000h	50.6.103/ 1758
F0	ECC Error Interrupt Flag and Enable Register (FR_EEIFER)	16	R/W	0000h	50.6.104/ 1759
F2	ECC Error Report and Injection Control Register (FR_EERICR)	16	R/W	0000h	50.6.105/ 1761
F4	ECC Error Report Address Register (FR_EERAR)	16	R	7000h	50.6.106/ 1762
F6	ECC Error Report Data Register (FR_EERDR)	16	R	0000h	50.6.107/ 1763
F8	ECC Error Report Code Register (FR_EERCR)	16	R	0000h	50.6.108/ 1764

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FA	ECC Error Injection Address Register (FR_EEIAR)	16	R/W	0000h	50.6.109/ 1765
FC	ECC Error Injection Data Register (FR_EEIDR)	16	R/W	0000h	50.6.110/ 1765
FE	ECC Error Injection Code Register (FR_EEICR)	16	R/W	0000h	50.6.111/ 1766
800	Message Buffer Configuration, Control, Status Register (FR_MBCCSR0)	16	R/W	0000h	50.6.112/ 1767
802	Message Buffer Cycle Counter Filter Register (FR_MBCCFR0)	16	R/W	See section	50.6.113/ 1768
804	Message Buffer Frame ID Register (FR_MBFIDR0)	16	R/W	See section	50.6.114/ 1770
806	Message Buffer Index Register (FR_MBIDXR0)	16	R/W	See section	50.6.115/ 1771
808	Message Buffer Configuration, Control, Status Register (FR_MBCCSR1)	16	R/W	0000h	50.6.112/ 1767
80A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR1)	16	R/W	See section	50.6.113/ 1768
80C	Message Buffer Frame ID Register (FR_MBFIDR1)	16	R/W	See section	50.6.114/ 1770
80E	Message Buffer Index Register (FR_MBIDXR1)	16	R/W	See section	50.6.115/ 1771
810	Message Buffer Configuration, Control, Status Register (FR_MBCCSR2)	16	R/W	0000h	50.6.112/ 1767
812	Message Buffer Cycle Counter Filter Register (FR_MBCCFR2)	16	R/W	See section	50.6.113/ 1768
814	Message Buffer Frame ID Register (FR_MBFIDR2)	16	R/W	See section	50.6.114/ 1770
816	Message Buffer Index Register (FR_MBIDXR2)	16	R/W	See section	50.6.115/ 1771
818	Message Buffer Configuration, Control, Status Register (FR_MBCCSR3)	16	R/W	0000h	50.6.112/ 1767
81A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR3)	16	R/W	See section	50.6.113/ 1768
81C	Message Buffer Frame ID Register (FR_MBFIDR3)	16	R/W	See section	50.6.114/ 1770
81E	Message Buffer Index Register (FR_MBIDXR3)	16	R/W	See section	50.6.115/ 1771
820	Message Buffer Configuration, Control, Status Register (FR_MBCCSR4)	16	R/W	0000h	50.6.112/ 1767
822	Message Buffer Cycle Counter Filter Register (FR_MBCCFR4)	16	R/W	See section	50.6.113/ 1768
824	Message Buffer Frame ID Register (FR_MBFIDR4)	16	R/W	See section	50.6.114/ 1770

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
826	Message Buffer Index Register (FR_MBIDXR4)	16	R/W	See section	50.6.115/ 1771
828	Message Buffer Configuration, Control, Status Register (FR_MBCCSR5)	16	R/W	0000h	50.6.112/ 1767
82A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR5)	16	R/W	See section	50.6.113/ 1768
82C	Message Buffer Frame ID Register (FR_MBFIDR5)	16	R/W	See section	50.6.114/ 1770
82E	Message Buffer Index Register (FR_MBIDXR5)	16	R/W	See section	50.6.115/ 1771
830	Message Buffer Configuration, Control, Status Register (FR_MBCCSR6)	16	R/W	0000h	50.6.112/ 1767
832	Message Buffer Cycle Counter Filter Register (FR_MBCCFR6)	16	R/W	See section	50.6.113/ 1768
834	Message Buffer Frame ID Register (FR_MBFIDR6)	16	R/W	See section	50.6.114/ 1770
836	Message Buffer Index Register (FR_MBIDXR6)	16	R/W	See section	50.6.115/ 1771
838	Message Buffer Configuration, Control, Status Register (FR_MBCCSR7)	16	R/W	0000h	50.6.112/ 1767
83A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR7)	16	R/W	See section	50.6.113/ 1768
83C	Message Buffer Frame ID Register (FR_MBFIDR7)	16	R/W	See section	50.6.114/ 1770
83E	Message Buffer Index Register (FR_MBIDXR7)	16	R/W	See section	50.6.115/ 1771
840	Message Buffer Configuration, Control, Status Register (FR_MBCCSR8)	16	R/W	0000h	50.6.112/ 1767
842	Message Buffer Cycle Counter Filter Register (FR_MBCCFR8)	16	R/W	See section	50.6.113/ 1768
844	Message Buffer Frame ID Register (FR_MBFIDR8)	16	R/W	See section	50.6.114/ 1770
846	Message Buffer Index Register (FR_MBIDXR8)	16	R/W	See section	50.6.115/ 1771
848	Message Buffer Configuration, Control, Status Register (FR_MBCCSR9)	16	R/W	0000h	50.6.112/ 1767
84A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR9)	16	R/W	See section	50.6.113/ 1768
84C	Message Buffer Frame ID Register (FR_MBFIDR9)	16	R/W	See section	50.6.114/ 1770
84E	Message Buffer Index Register (FR_MBIDXR9)	16	R/W	See section	50.6.115/ 1771
850	Message Buffer Configuration, Control, Status Register (FR_MBCCSR10)	16	R/W	0000h	50.6.112/ 1767

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
852	Message Buffer Cycle Counter Filter Register (FR_MBCCFR10)	16	R/W	See section	50.6.113/ 1768
854	Message Buffer Frame ID Register (FR_MBFIDR10)	16	R/W	See section	50.6.114/ 1770
856	Message Buffer Index Register (FR_MBIDXR10)	16	R/W	See section	50.6.115/ 1771
858	Message Buffer Configuration, Control, Status Register (FR_MBCCSR11)	16	R/W	0000h	50.6.112/ 1767
85A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR11)	16	R/W	See section	50.6.113/ 1768
85C	Message Buffer Frame ID Register (FR_MBFIDR11)	16	R/W	See section	50.6.114/ 1770
85E	Message Buffer Index Register (FR_MBIDXR11)	16	R/W	See section	50.6.115/ 1771
860	Message Buffer Configuration, Control, Status Register (FR_MBCCSR12)	16	R/W	0000h	50.6.112/ 1767
862	Message Buffer Cycle Counter Filter Register (FR_MBCCFR12)	16	R/W	See section	50.6.113/ 1768
864	Message Buffer Frame ID Register (FR_MBFIDR12)	16	R/W	See section	50.6.114/ 1770
866	Message Buffer Index Register (FR_MBIDXR12)	16	R/W	See section	50.6.115/ 1771
868	Message Buffer Configuration, Control, Status Register (FR_MBCCSR13)	16	R/W	0000h	50.6.112/ 1767
86A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR13)	16	R/W	See section	50.6.113/ 1768
86C	Message Buffer Frame ID Register (FR_MBFIDR13)	16	R/W	See section	50.6.114/ 1770
86E	Message Buffer Index Register (FR_MBIDXR13)	16	R/W	See section	50.6.115/ 1771
870	Message Buffer Configuration, Control, Status Register (FR_MBCCSR14)	16	R/W	0000h	50.6.112/ 1767
872	Message Buffer Cycle Counter Filter Register (FR_MBCCFR14)	16	R/W	See section	50.6.113/ 1768
874	Message Buffer Frame ID Register (FR_MBFIDR14)	16	R/W	See section	50.6.114/ 1770
876	Message Buffer Index Register (FR_MBIDXR14)	16	R/W	See section	50.6.115/ 1771
878	Message Buffer Configuration, Control, Status Register (FR_MBCCSR15)	16	R/W	0000h	50.6.112/ 1767
87A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR15)	16	R/W	See section	50.6.113/ 1768
87C	Message Buffer Frame ID Register (FR_MBFIDR15)	16	R/W	See section	50.6.114/ 1770

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
87E	Message Buffer Index Register (FR_MBIDXR15)	16	R/W	See section	50.6.115/ 1771
880	Message Buffer Configuration, Control, Status Register (FR_MBCCSR16)	16	R/W	0000h	50.6.112/ 1767
882	Message Buffer Cycle Counter Filter Register (FR_MBCCFR16)	16	R/W	See section	50.6.113/ 1768
884	Message Buffer Frame ID Register (FR_MBFIDR16)	16	R/W	See section	50.6.114/ 1770
886	Message Buffer Index Register (FR_MBIDXR16)	16	R/W	See section	50.6.115/ 1771
888	Message Buffer Configuration, Control, Status Register (FR_MBCCSR17)	16	R/W	0000h	50.6.112/ 1767
88A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR17)	16	R/W	See section	50.6.113/ 1768
88C	Message Buffer Frame ID Register (FR_MBFIDR17)	16	R/W	See section	50.6.114/ 1770
88E	Message Buffer Index Register (FR_MBIDXR17)	16	R/W	See section	50.6.115/ 1771
890	Message Buffer Configuration, Control, Status Register (FR_MBCCSR18)	16	R/W	0000h	50.6.112/ 1767
892	Message Buffer Cycle Counter Filter Register (FR_MBCCFR18)	16	R/W	See section	50.6.113/ 1768
894	Message Buffer Frame ID Register (FR_MBFIDR18)	16	R/W	See section	50.6.114/ 1770
896	Message Buffer Index Register (FR_MBIDXR18)	16	R/W	See section	50.6.115/ 1771
898	Message Buffer Configuration, Control, Status Register (FR_MBCCSR19)	16	R/W	0000h	50.6.112/ 1767
89A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR19)	16	R/W	See section	50.6.113/ 1768
89C	Message Buffer Frame ID Register (FR_MBFIDR19)	16	R/W	See section	50.6.114/ 1770
89E	Message Buffer Index Register (FR_MBIDXR19)	16	R/W	See section	50.6.115/ 1771
8A0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR20)	16	R/W	0000h	50.6.112/ 1767
8A2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR20)	16	R/W	See section	50.6.113/ 1768
8A4	Message Buffer Frame ID Register (FR_MBFIDR20)	16	R/W	See section	50.6.114/ 1770
8A6	Message Buffer Index Register (FR_MBIDXR20)	16	R/W	See section	50.6.115/ 1771
8A8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR21)	16	R/W	0000h	50.6.112/ 1767

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8AA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR21)	16	R/W	See section	50.6.113/ 1768
8AC	Message Buffer Frame ID Register (FR_MBFIDR21)	16	R/W	See section	50.6.114/ 1770
8AE	Message Buffer Index Register (FR_MBIDXR21)	16	R/W	See section	50.6.115/ 1771
8B0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR22)	16	R/W	0000h	50.6.112/ 1767
8B2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR22)	16	R/W	See section	50.6.113/ 1768
8B4	Message Buffer Frame ID Register (FR_MBFIDR22)	16	R/W	See section	50.6.114/ 1770
8B6	Message Buffer Index Register (FR_MBIDXR22)	16	R/W	See section	50.6.115/ 1771
8B8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR23)	16	R/W	0000h	50.6.112/ 1767
8BA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR23)	16	R/W	See section	50.6.113/ 1768
8BC	Message Buffer Frame ID Register (FR_MBFIDR23)	16	R/W	See section	50.6.114/ 1770
8BE	Message Buffer Index Register (FR_MBIDXR23)	16	R/W	See section	50.6.115/ 1771
8C0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR24)	16	R/W	0000h	50.6.112/ 1767
8C2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR24)	16	R/W	See section	50.6.113/ 1768
8C4	Message Buffer Frame ID Register (FR_MBFIDR24)	16	R/W	See section	50.6.114/ 1770
8C6	Message Buffer Index Register (FR_MBIDXR24)	16	R/W	See section	50.6.115/ 1771
8C8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR25)	16	R/W	0000h	50.6.112/ 1767
8CA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR25)	16	R/W	See section	50.6.113/ 1768
8CC	Message Buffer Frame ID Register (FR_MBFIDR25)	16	R/W	See section	50.6.114/ 1770
8CE	Message Buffer Index Register (FR_MBIDXR25)	16	R/W	See section	50.6.115/ 1771
8D0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR26)	16	R/W	0000h	50.6.112/ 1767
8D2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR26)	16	R/W	See section	50.6.113/ 1768
8D4	Message Buffer Frame ID Register (FR_MBFIDR26)	16	R/W	See section	50.6.114/ 1770

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8D6	Message Buffer Index Register (FR_MBIDXR26)	16	R/W	See section	50.6.115/ 1771
8D8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR27)	16	R/W	0000h	50.6.112/ 1767
8DA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR27)	16	R/W	See section	50.6.113/ 1768
8DC	Message Buffer Frame ID Register (FR_MBFIDR27)	16	R/W	See section	50.6.114/ 1770
8DE	Message Buffer Index Register (FR_MBIDXR27)	16	R/W	See section	50.6.115/ 1771
8E0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR28)	16	R/W	0000h	50.6.112/ 1767
8E2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR28)	16	R/W	See section	50.6.113/ 1768
8E4	Message Buffer Frame ID Register (FR_MBFIDR28)	16	R/W	See section	50.6.114/ 1770
8E6	Message Buffer Index Register (FR_MBIDXR28)	16	R/W	See section	50.6.115/ 1771
8E8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR29)	16	R/W	0000h	50.6.112/ 1767
8EA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR29)	16	R/W	See section	50.6.113/ 1768
8EC	Message Buffer Frame ID Register (FR_MBFIDR29)	16	R/W	See section	50.6.114/ 1770
8EE	Message Buffer Index Register (FR_MBIDXR29)	16	R/W	See section	50.6.115/ 1771
8F0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR30)	16	R/W	0000h	50.6.112/ 1767
8F2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR30)	16	R/W	See section	50.6.113/ 1768
8F4	Message Buffer Frame ID Register (FR_MBFIDR30)	16	R/W	See section	50.6.114/ 1770
8F6	Message Buffer Index Register (FR_MBIDXR30)	16	R/W	See section	50.6.115/ 1771
8F8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR31)	16	R/W	0000h	50.6.112/ 1767
8FA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR31)	16	R/W	See section	50.6.113/ 1768
8FC	Message Buffer Frame ID Register (FR_MBFIDR31)	16	R/W	See section	50.6.114/ 1770
8FE	Message Buffer Index Register (FR_MBIDXR31)	16	R/W	See section	50.6.115/ 1771
900	Message Buffer Configuration, Control, Status Register (FR_MBCCSR32)	16	R/W	0000h	50.6.112/ 1767

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
902	Message Buffer Cycle Counter Filter Register (FR_MBCCFR32)	16	R/W	See section	50.6.113/ 1768
904	Message Buffer Frame ID Register (FR_MBFIDR32)	16	R/W	See section	50.6.114/ 1770
906	Message Buffer Index Register (FR_MBIDXR32)	16	R/W	See section	50.6.115/ 1771
908	Message Buffer Configuration, Control, Status Register (FR_MBCCSR33)	16	R/W	0000h	50.6.112/ 1767
90A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR33)	16	R/W	See section	50.6.113/ 1768
90C	Message Buffer Frame ID Register (FR_MBFIDR33)	16	R/W	See section	50.6.114/ 1770
90E	Message Buffer Index Register (FR_MBIDXR33)	16	R/W	See section	50.6.115/ 1771
910	Message Buffer Configuration, Control, Status Register (FR_MBCCSR34)	16	R/W	0000h	50.6.112/ 1767
912	Message Buffer Cycle Counter Filter Register (FR_MBCCFR34)	16	R/W	See section	50.6.113/ 1768
914	Message Buffer Frame ID Register (FR_MBFIDR34)	16	R/W	See section	50.6.114/ 1770
916	Message Buffer Index Register (FR_MBIDXR34)	16	R/W	See section	50.6.115/ 1771
918	Message Buffer Configuration, Control, Status Register (FR_MBCCSR35)	16	R/W	0000h	50.6.112/ 1767
91A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR35)	16	R/W	See section	50.6.113/ 1768
91C	Message Buffer Frame ID Register (FR_MBFIDR35)	16	R/W	See section	50.6.114/ 1770
91E	Message Buffer Index Register (FR_MBIDXR35)	16	R/W	See section	50.6.115/ 1771
920	Message Buffer Configuration, Control, Status Register (FR_MBCCSR36)	16	R/W	0000h	50.6.112/ 1767
922	Message Buffer Cycle Counter Filter Register (FR_MBCCFR36)	16	R/W	See section	50.6.113/ 1768
924	Message Buffer Frame ID Register (FR_MBFIDR36)	16	R/W	See section	50.6.114/ 1770
926	Message Buffer Index Register (FR_MBIDXR36)	16	R/W	See section	50.6.115/ 1771
928	Message Buffer Configuration, Control, Status Register (FR_MBCCSR37)	16	R/W	0000h	50.6.112/ 1767
92A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR37)	16	R/W	See section	50.6.113/ 1768
92C	Message Buffer Frame ID Register (FR_MBFIDR37)	16	R/W	See section	50.6.114/ 1770

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
92E	Message Buffer Index Register (FR_MBIDXR37)	16	R/W	See section	50.6.115/ 1771
930	Message Buffer Configuration, Control, Status Register (FR_MBCCSR38)	16	R/W	0000h	50.6.112/ 1767
932	Message Buffer Cycle Counter Filter Register (FR_MBCCFR38)	16	R/W	See section	50.6.113/ 1768
934	Message Buffer Frame ID Register (FR_MBFIDR38)	16	R/W	See section	50.6.114/ 1770
936	Message Buffer Index Register (FR_MBIDXR38)	16	R/W	See section	50.6.115/ 1771
938	Message Buffer Configuration, Control, Status Register (FR_MBCCSR39)	16	R/W	0000h	50.6.112/ 1767
93A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR39)	16	R/W	See section	50.6.113/ 1768
93C	Message Buffer Frame ID Register (FR_MBFIDR39)	16	R/W	See section	50.6.114/ 1770
93E	Message Buffer Index Register (FR_MBIDXR39)	16	R/W	See section	50.6.115/ 1771
940	Message Buffer Configuration, Control, Status Register (FR_MBCCSR40)	16	R/W	0000h	50.6.112/ 1767
942	Message Buffer Cycle Counter Filter Register (FR_MBCCFR40)	16	R/W	See section	50.6.113/ 1768
944	Message Buffer Frame ID Register (FR_MBFIDR40)	16	R/W	See section	50.6.114/ 1770
946	Message Buffer Index Register (FR_MBIDXR40)	16	R/W	See section	50.6.115/ 1771
948	Message Buffer Configuration, Control, Status Register (FR_MBCCSR41)	16	R/W	0000h	50.6.112/ 1767
94A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR41)	16	R/W	See section	50.6.113/ 1768
94C	Message Buffer Frame ID Register (FR_MBFIDR41)	16	R/W	See section	50.6.114/ 1770
94E	Message Buffer Index Register (FR_MBIDXR41)	16	R/W	See section	50.6.115/ 1771
950	Message Buffer Configuration, Control, Status Register (FR_MBCCSR42)	16	R/W	0000h	50.6.112/ 1767
952	Message Buffer Cycle Counter Filter Register (FR_MBCCFR42)	16	R/W	See section	50.6.113/ 1768
954	Message Buffer Frame ID Register (FR_MBFIDR42)	16	R/W	See section	50.6.114/ 1770
956	Message Buffer Index Register (FR_MBIDXR42)	16	R/W	See section	50.6.115/ 1771
958	Message Buffer Configuration, Control, Status Register (FR_MBCCSR43)	16	R/W	0000h	50.6.112/ 1767

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
95A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR43)	16	R/W	See section	50.6.113/ 1768
95C	Message Buffer Frame ID Register (FR_MBFIDR43)	16	R/W	See section	50.6.114/ 1770
95E	Message Buffer Index Register (FR_MBIDXR43)	16	R/W	See section	50.6.115/ 1771
960	Message Buffer Configuration, Control, Status Register (FR_MBCCSR44)	16	R/W	0000h	50.6.112/ 1767
962	Message Buffer Cycle Counter Filter Register (FR_MBCCFR44)	16	R/W	See section	50.6.113/ 1768
964	Message Buffer Frame ID Register (FR_MBFIDR44)	16	R/W	See section	50.6.114/ 1770
966	Message Buffer Index Register (FR_MBIDXR44)	16	R/W	See section	50.6.115/ 1771
968	Message Buffer Configuration, Control, Status Register (FR_MBCCSR45)	16	R/W	0000h	50.6.112/ 1767
96A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR45)	16	R/W	See section	50.6.113/ 1768
96C	Message Buffer Frame ID Register (FR_MBFIDR45)	16	R/W	See section	50.6.114/ 1770
96E	Message Buffer Index Register (FR_MBIDXR45)	16	R/W	See section	50.6.115/ 1771
970	Message Buffer Configuration, Control, Status Register (FR_MBCCSR46)	16	R/W	0000h	50.6.112/ 1767
972	Message Buffer Cycle Counter Filter Register (FR_MBCCFR46)	16	R/W	See section	50.6.113/ 1768
974	Message Buffer Frame ID Register (FR_MBFIDR46)	16	R/W	See section	50.6.114/ 1770
976	Message Buffer Index Register (FR_MBIDXR46)	16	R/W	See section	50.6.115/ 1771
978	Message Buffer Configuration, Control, Status Register (FR_MBCCSR47)	16	R/W	0000h	50.6.112/ 1767
97A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR47)	16	R/W	See section	50.6.113/ 1768
97C	Message Buffer Frame ID Register (FR_MBFIDR47)	16	R/W	See section	50.6.114/ 1770
97E	Message Buffer Index Register (FR_MBIDXR47)	16	R/W	See section	50.6.115/ 1771
980	Message Buffer Configuration, Control, Status Register (FR_MBCCSR48)	16	R/W	0000h	50.6.112/ 1767
982	Message Buffer Cycle Counter Filter Register (FR_MBCCFR48)	16	R/W	See section	50.6.113/ 1768
984	Message Buffer Frame ID Register (FR_MBFIDR48)	16	R/W	See section	50.6.114/ 1770

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
986	Message Buffer Index Register (FR_MBIDXR48)	16	R/W	See section	50.6.115/ 1771
988	Message Buffer Configuration, Control, Status Register (FR_MBCCSR49)	16	R/W	0000h	50.6.112/ 1767
98A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR49)	16	R/W	See section	50.6.113/ 1768
98C	Message Buffer Frame ID Register (FR_MBFIDR49)	16	R/W	See section	50.6.114/ 1770
98E	Message Buffer Index Register (FR_MBIDXR49)	16	R/W	See section	50.6.115/ 1771
990	Message Buffer Configuration, Control, Status Register (FR_MBCCSR50)	16	R/W	0000h	50.6.112/ 1767
992	Message Buffer Cycle Counter Filter Register (FR_MBCCFR50)	16	R/W	See section	50.6.113/ 1768
994	Message Buffer Frame ID Register (FR_MBFIDR50)	16	R/W	See section	50.6.114/ 1770
996	Message Buffer Index Register (FR_MBIDXR50)	16	R/W	See section	50.6.115/ 1771
998	Message Buffer Configuration, Control, Status Register (FR_MBCCSR51)	16	R/W	0000h	50.6.112/ 1767
99A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR51)	16	R/W	See section	50.6.113/ 1768
99C	Message Buffer Frame ID Register (FR_MBFIDR51)	16	R/W	See section	50.6.114/ 1770
99E	Message Buffer Index Register (FR_MBIDXR51)	16	R/W	See section	50.6.115/ 1771
9A0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR52)	16	R/W	0000h	50.6.112/ 1767
9A2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR52)	16	R/W	See section	50.6.113/ 1768
9A4	Message Buffer Frame ID Register (FR_MBFIDR52)	16	R/W	See section	50.6.114/ 1770
9A6	Message Buffer Index Register (FR_MBIDXR52)	16	R/W	See section	50.6.115/ 1771
9A8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR53)	16	R/W	0000h	50.6.112/ 1767
9AA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR53)	16	R/W	See section	50.6.113/ 1768
9AC	Message Buffer Frame ID Register (FR_MBFIDR53)	16	R/W	See section	50.6.114/ 1770
9AE	Message Buffer Index Register (FR_MBIDXR53)	16	R/W	See section	50.6.115/ 1771
9B0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR54)	16	R/W	0000h	50.6.112/ 1767

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9B2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR54)	16	R/W	See section	50.6.113/ 1768
9B4	Message Buffer Frame ID Register (FR_MBFIDR54)	16	R/W	See section	50.6.114/ 1770
9B6	Message Buffer Index Register (FR_MBIDXR54)	16	R/W	See section	50.6.115/ 1771
9B8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR55)	16	R/W	0000h	50.6.112/ 1767
9BA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR55)	16	R/W	See section	50.6.113/ 1768
9BC	Message Buffer Frame ID Register (FR_MBFIDR55)	16	R/W	See section	50.6.114/ 1770
9BE	Message Buffer Index Register (FR_MBIDXR55)	16	R/W	See section	50.6.115/ 1771
9C0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR56)	16	R/W	0000h	50.6.112/ 1767
9C2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR56)	16	R/W	See section	50.6.113/ 1768
9C4	Message Buffer Frame ID Register (FR_MBFIDR56)	16	R/W	See section	50.6.114/ 1770
9C6	Message Buffer Index Register (FR_MBIDXR56)	16	R/W	See section	50.6.115/ 1771
9C8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR57)	16	R/W	0000h	50.6.112/ 1767
9CA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR57)	16	R/W	See section	50.6.113/ 1768
9CC	Message Buffer Frame ID Register (FR_MBFIDR57)	16	R/W	See section	50.6.114/ 1770
9CE	Message Buffer Index Register (FR_MBIDXR57)	16	R/W	See section	50.6.115/ 1771
9D0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR58)	16	R/W	0000h	50.6.112/ 1767
9D2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR58)	16	R/W	See section	50.6.113/ 1768
9D4	Message Buffer Frame ID Register (FR_MBFIDR58)	16	R/W	See section	50.6.114/ 1770
9D6	Message Buffer Index Register (FR_MBIDXR58)	16	R/W	See section	50.6.115/ 1771
9D8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR59)	16	R/W	0000h	50.6.112/ 1767
9DA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR59)	16	R/W	See section	50.6.113/ 1768
9DC	Message Buffer Frame ID Register (FR_MBFIDR59)	16	R/W	See section	50.6.114/ 1770

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9DE	Message Buffer Index Register (FR_MBIDXR59)	16	R/W	See section	50.6.115/ 1771
9E0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR60)	16	R/W	0000h	50.6.112/ 1767
9E2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR60)	16	R/W	See section	50.6.113/ 1768
9E4	Message Buffer Frame ID Register (FR_MBFIDR60)	16	R/W	See section	50.6.114/ 1770
9E6	Message Buffer Index Register (FR_MBIDXR60)	16	R/W	See section	50.6.115/ 1771
9E8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR61)	16	R/W	0000h	50.6.112/ 1767
9EA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR61)	16	R/W	See section	50.6.113/ 1768
9EC	Message Buffer Frame ID Register (FR_MBFIDR61)	16	R/W	See section	50.6.114/ 1770
9EE	Message Buffer Index Register (FR_MBIDXR61)	16	R/W	See section	50.6.115/ 1771
9F0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR62)	16	R/W	0000h	50.6.112/ 1767
9F2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR62)	16	R/W	See section	50.6.113/ 1768
9F4	Message Buffer Frame ID Register (FR_MBFIDR62)	16	R/W	See section	50.6.114/ 1770
9F6	Message Buffer Index Register (FR_MBIDXR62)	16	R/W	See section	50.6.115/ 1771
9F8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR63)	16	R/W	0000h	50.6.112/ 1767
9FA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR63)	16	R/W	See section	50.6.113/ 1768
9FC	Message Buffer Frame ID Register (FR_MBFIDR63)	16	R/W	See section	50.6.114/ 1770
9FE	Message Buffer Index Register (FR_MBIDXR63)	16	R/W	See section	50.6.115/ 1771
1000	Message Buffer Data Field Offset Register (FR_MBDOR0)	16	R/W	See section	50.6.116/ 1772
1002	Message Buffer Data Field Offset Register (FR_MBDOR1)	16	R/W	See section	50.6.116/ 1772
1004	Message Buffer Data Field Offset Register (FR_MBDOR2)	16	R/W	See section	50.6.116/ 1772
1006	Message Buffer Data Field Offset Register (FR_MBDOR3)	16	R/W	See section	50.6.116/ 1772
1008	Message Buffer Data Field Offset Register (FR_MBDOR4)	16	R/W	See section	50.6.116/ 1772

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
100A	Message Buffer Data Field Offset Register (FR_MBDOR5)	16	R/W	See section	50.6.116/ 1772
100C	Message Buffer Data Field Offset Register (FR_MBDOR6)	16	R/W	See section	50.6.116/ 1772
100E	Message Buffer Data Field Offset Register (FR_MBDOR7)	16	R/W	See section	50.6.116/ 1772
1010	Message Buffer Data Field Offset Register (FR_MBDOR8)	16	R/W	See section	50.6.116/ 1772
1012	Message Buffer Data Field Offset Register (FR_MBDOR9)	16	R/W	See section	50.6.116/ 1772
1014	Message Buffer Data Field Offset Register (FR_MBDOR10)	16	R/W	See section	50.6.116/ 1772
1016	Message Buffer Data Field Offset Register (FR_MBDOR11)	16	R/W	See section	50.6.116/ 1772
1018	Message Buffer Data Field Offset Register (FR_MBDOR12)	16	R/W	See section	50.6.116/ 1772
101A	Message Buffer Data Field Offset Register (FR_MBDOR13)	16	R/W	See section	50.6.116/ 1772
101C	Message Buffer Data Field Offset Register (FR_MBDOR14)	16	R/W	See section	50.6.116/ 1772
101E	Message Buffer Data Field Offset Register (FR_MBDOR15)	16	R/W	See section	50.6.116/ 1772
1020	Message Buffer Data Field Offset Register (FR_MBDOR16)	16	R/W	See section	50.6.116/ 1772
1022	Message Buffer Data Field Offset Register (FR_MBDOR17)	16	R/W	See section	50.6.116/ 1772
1024	Message Buffer Data Field Offset Register (FR_MBDOR18)	16	R/W	See section	50.6.116/ 1772
1026	Message Buffer Data Field Offset Register (FR_MBDOR19)	16	R/W	See section	50.6.116/ 1772
1028	Message Buffer Data Field Offset Register (FR_MBDOR20)	16	R/W	See section	50.6.116/ 1772
102A	Message Buffer Data Field Offset Register (FR_MBDOR21)	16	R/W	See section	50.6.116/ 1772
102C	Message Buffer Data Field Offset Register (FR_MBDOR22)	16	R/W	See section	50.6.116/ 1772
102E	Message Buffer Data Field Offset Register (FR_MBDOR23)	16	R/W	See section	50.6.116/ 1772
1030	Message Buffer Data Field Offset Register (FR_MBDOR24)	16	R/W	See section	50.6.116/ 1772
1032	Message Buffer Data Field Offset Register (FR_MBDOR25)	16	R/W	See section	50.6.116/ 1772
1034	Message Buffer Data Field Offset Register (FR_MBDOR26)	16	R/W	See section	50.6.116/ 1772

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1036	Message Buffer Data Field Offset Register (FR_MBDOR27)	16	R/W	See section	50.6.116/ 1772
1038	Message Buffer Data Field Offset Register (FR_MBDOR28)	16	R/W	See section	50.6.116/ 1772
103A	Message Buffer Data Field Offset Register (FR_MBDOR29)	16	R/W	See section	50.6.116/ 1772
103C	Message Buffer Data Field Offset Register (FR_MBDOR30)	16	R/W	See section	50.6.116/ 1772
103E	Message Buffer Data Field Offset Register (FR_MBDOR31)	16	R/W	See section	50.6.116/ 1772
1040	Message Buffer Data Field Offset Register (FR_MBDOR32)	16	R/W	See section	50.6.116/ 1772
1042	Message Buffer Data Field Offset Register (FR_MBDOR33)	16	R/W	See section	50.6.116/ 1772
1044	Message Buffer Data Field Offset Register (FR_MBDOR34)	16	R/W	See section	50.6.116/ 1772
1046	Message Buffer Data Field Offset Register (FR_MBDOR35)	16	R/W	See section	50.6.116/ 1772
1048	Message Buffer Data Field Offset Register (FR_MBDOR36)	16	R/W	See section	50.6.116/ 1772
104A	Message Buffer Data Field Offset Register (FR_MBDOR37)	16	R/W	See section	50.6.116/ 1772
104C	Message Buffer Data Field Offset Register (FR_MBDOR38)	16	R/W	See section	50.6.116/ 1772
104E	Message Buffer Data Field Offset Register (FR_MBDOR39)	16	R/W	See section	50.6.116/ 1772
1050	Message Buffer Data Field Offset Register (FR_MBDOR40)	16	R/W	See section	50.6.116/ 1772
1052	Message Buffer Data Field Offset Register (FR_MBDOR41)	16	R/W	See section	50.6.116/ 1772
1054	Message Buffer Data Field Offset Register (FR_MBDOR42)	16	R/W	See section	50.6.116/ 1772
1056	Message Buffer Data Field Offset Register (FR_MBDOR43)	16	R/W	See section	50.6.116/ 1772
1058	Message Buffer Data Field Offset Register (FR_MBDOR44)	16	R/W	See section	50.6.116/ 1772
105A	Message Buffer Data Field Offset Register (FR_MBDOR45)	16	R/W	See section	50.6.116/ 1772
105C	Message Buffer Data Field Offset Register (FR_MBDOR46)	16	R/W	See section	50.6.116/ 1772
105E	Message Buffer Data Field Offset Register (FR_MBDOR47)	16	R/W	See section	50.6.116/ 1772
1060	Message Buffer Data Field Offset Register (FR_MBDOR48)	16	R/W	See section	50.6.116/ 1772

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1062	Message Buffer Data Field Offset Register (FR_MBDOR49)	16	R/W	See section	50.6.116/ 1772
1064	Message Buffer Data Field Offset Register (FR_MBDOR50)	16	R/W	See section	50.6.116/ 1772
1066	Message Buffer Data Field Offset Register (FR_MBDOR51)	16	R/W	See section	50.6.116/ 1772
1068	Message Buffer Data Field Offset Register (FR_MBDOR52)	16	R/W	See section	50.6.116/ 1772
106A	Message Buffer Data Field Offset Register (FR_MBDOR53)	16	R/W	See section	50.6.116/ 1772
106C	Message Buffer Data Field Offset Register (FR_MBDOR54)	16	R/W	See section	50.6.116/ 1772
106E	Message Buffer Data Field Offset Register (FR_MBDOR55)	16	R/W	See section	50.6.116/ 1772
1070	Message Buffer Data Field Offset Register (FR_MBDOR56)	16	R/W	See section	50.6.116/ 1772
1072	Message Buffer Data Field Offset Register (FR_MBDOR57)	16	R/W	See section	50.6.116/ 1772
1074	Message Buffer Data Field Offset Register (FR_MBDOR58)	16	R/W	See section	50.6.116/ 1772
1076	Message Buffer Data Field Offset Register (FR_MBDOR59)	16	R/W	See section	50.6.116/ 1772
1078	Message Buffer Data Field Offset Register (FR_MBDOR60)	16	R/W	See section	50.6.116/ 1772
107A	Message Buffer Data Field Offset Register (FR_MBDOR61)	16	R/W	See section	50.6.116/ 1772
107C	Message Buffer Data Field Offset Register (FR_MBDOR62)	16	R/W	See section	50.6.116/ 1772
107E	Message Buffer Data Field Offset Register (FR_MBDOR63)	16	R/W	See section	50.6.116/ 1772
1080	Message Buffer Data Field Offset Register (FR_MBDOR64)	16	R/W	See section	50.6.116/ 1772
1082	Message Buffer Data Field Offset Register (FR_MBDOR65)	16	R/W	See section	50.6.116/ 1772
1084	Message Buffer Data Field Offset Register (FR_MBDOR66)	16	R/W	See section	50.6.116/ 1772
1086	Message Buffer Data Field Offset Register (FR_MBDOR67)	16	R/W	See section	50.6.116/ 1772
1090	LRAM ECC Error Test Register (FR_LEETR0)	16	R/W	See section	50.6.117/ 1772
1092	LRAM ECC Error Test Register (FR_LEETR1)	16	R/W	See section	50.6.117/ 1772
1094	LRAM ECC Error Test Register (FR_LEETR2)	16	R/W	See section	50.6.117/ 1772

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1096	LRAM ECC Error Test Register (FR_LEETR3)	16	R/W	See section	50.6.117/ 1772
1098	LRAM ECC Error Test Register (FR_LEETR4)	16	R/W	See section	50.6.117/ 1772
109A	LRAM ECC Error Test Register (FR_LEETR5)	16	R/W	See section	50.6.117/ 1772

50.6.1 Module Version Register (FR_MVR)

This register provides the CC version number. The module version number is derived from the CHI version number and the PE version number.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHIVER								PEVER							
Write																
Reset	1	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0

FR_MVR field descriptions

Field	Description
0–7 CHIVER	CHI Version Number. This field provides the version number of the controller host interface.
8–15 PEVER	PE Version Number. This field provides the version number of the protocol engine.

50.6.2 Module Configuration Register (FR_MCR)

This register defines the global configuration of the CC.

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read Write	MEN	SBFF	SCM	CHB	CHA	SFFE	ECCE	Reserved
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	FUM	FAM	0	CLKSEL	BITRATE			0
Write								
Reset	0	0	0	0	0	0	0	0

FR_MCR field descriptions

Field	Description
0 MEN	Module Enable. This bit indicates whether or not the CC is in the Disabled Mode. The application requests the CC to leave the Disabled Mode by writing 1 to this bit Before leaving the Disabled Mode, the application must configure the SCM, SBFF, CHB, CHA, TMODE, BITRATE values. For details see Modes of Operation. **NOTE:** If the CC is enabled it can only be disabled during mode: POC:default config. 0 Write: only during POC:default config, CC disable Read: CC disabled 1 Write: enable CC Read: CC enabled
1 SBFF	System Bus Failure Freeze This bit controls the behavior of the CC in case of a system bus failure. 0 Continue normal operation 1 Transition to freeze mode
2 SCM	Single Channel Device Mode. This control bit defines the channel device mode of the CC as described in Channel Device Modes. 0 CC works in dual channel device mode 1 CC works in single channel device mode
3 CHB	Channel B Enable. protocol related parameter: pChannels The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given in the following table.

Table 50-7. FlexRay channel selection

SCM	CHB	CHA	Description
Dual Channel Device Modes			
0	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A

Table continues on the next page...

FR_MCR field descriptions (continued)

Field	Description

Table 50-7. FlexRay channel selection (continued)

SCM	CHB	CHA	Description
			ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	1	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel B
	1	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel B
Single Channel Device Mode			
1	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	1	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ driven by CC - connected to FlexRay channel B ports FR_B_RX, FR_B_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ not driven by CC
	1	1	reserved

Field	Description
4 CHA	Channel A Enable. protocol related parameter: pChannels The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given in the FlexRay channel selection table as seen above.
5 SFFE	Synchronization Frame Filter Enable. This bit controls the filtering for received synchronization frames. For details see Sync Frame Filtering . 0 Synchronization frame filtering disabled 1 Synchronization frame filtering enabled
6 ECCE	ECC Functionality Enable. This bit controls the ECC memory error detection functionality. For details see Memory Content Fault Detection . 0 ECC functionality (injection, detection, reporting, response) disabled 1 ECC functionality enabled
7 Reserved	Reserved bit, will not be changed. Application must not write any value different from the reset value. This field is reserved.
8 FUM	FIFO Update Mode. This bit controls the FIFO update behavior when the interrupt flags FR_GIFER[FAFAIF] and FR_GIFER[FAFBIF] are written by the application (see FIFO Update) 0 FIFOA/FIFOB is updated on writing 1 to FR_GIFER[FAFAIF] /FR_GIFER[FAFBIF] 1 FIFOA/FIFOB) is not updated on writing 1 to FR_GIFER[FAFAIF]/FR_GIFER[FAFBIF]
9 FAM	FIFO Address Mode. This bit controls the location of the system memory base address for the FIFOs. (see FIFO Configuration)

Table continues on the next page...

FR_MCR field descriptions (continued)

Field	Description
	0 FIFO Base Address located in System Memory Base Address High Register (FR_SYMBADHR)) 1 FIFO Base Address located in Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 CLKSEL	Protocol Engine Clock Source Select. This bit is used to select the clock source for the protocol engine. 0 PE clock source is generated by on-chip crystal oscillator. 1 PE clock source is generated by on-chip PLL.
12–14 BITRATE	FlexRay Bus Bit Rate. This bit field defines the FlexRay Bus Bit Rate. 000 10.0 Mbit/sec 001 5.0 Mbit/sec 010 2.5 Mbit/sec 011 8.0 Mbit/sec 100 reserved 101 reserved 110 reserved 111 reserved
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

50.6.3 System Memory Base Address High Register (FR_SYMBADHR)

NOTE

> The system memory base address must be set before the CC is enabled.

The system memory base address registers (FR_SYMBADHR) define the base address of the FlexRay memory area within the system memory. The base address is used by the BMIF to calculate the physical memory address for system memory accesses.

The FR_SYMBADHR register contains the most significant bits of the base address.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SYMBADHR field descriptions

Field	Description
0–15 SMBA	System Memory Base Address high. This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.

50.6.4 System Memory Base Address Low Register (FR_SYMBADLR)

NOTE

> The system memory base address must be set before the CC is enabled.

The system memory base address registers (FR_SYMBADLR) define the base address of the FlexRay memory area within the system memory. The base address is used by the BMIF to calculate the physical memory address for system memory accesses.

The FR_SYMBADLR register contains the least significant bits of the base address.

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA												0 ▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SYMBADLR field descriptions

Field	Description
0–11 SMBA	System Memory Base Address low. This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

50.6.5 Strobe Signal Control Register (FR_STBSCR)

This register is used to assign the individual protocol timing related strobe signals given in Table 50-8 to the external strobe ports. Each strobe signal can be assigned to at most one strobe port . Each write access to registers overwrites the previously written ENB and STBPSEL values for the signal indicated by SEL . If more than one strobe signal is assigned to one strobe port, the current values of the strobe signals are combined with a binary OR and presented at the strobe port . If no strobe signal is assigned to a strobe port, the strobe port carries logic 0 . For more detailed and timing information refer to Strobe Signal Support .

NOTE

In single channel device mode, channel B related strobe signals are undefined and should not be assigned to the strobe ports.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0			SEL				0			ENB	0		STBPSEL	
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_STBSCR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SEL	Strobe Signal Select. This control field selects one of the strobe signals given in Table 50-8 to be enabled or disabled and assigned to one of the four strobe ports given in the following table.

Table 50-8. Strobe Signal Mapping

SEL		Description	Channel	Type	Offset (Given in PE clock cycles)	Reference
dec	hex					
0	0x0	arm	-	value	+1	MT start
1	0x1	mt	-	value	+1	MT start
2	0x2	cycle start	-	pulse	0	MT start
3	0x3	minislot start	-	pulse	0	MT start
4	0x4	slot start	A	pulse	0	MT start
5	0x5		B			
6	0x6	receive data after glitch filtering	A	value	+4	FR_A_RX
7	0x7		B			FR_B_RX
8	0x8	channel idle indicator	A	level	+5	FR_A_RX
9	0x9		B			FR_B_RX
10	0xA	syntax error detected	A	pulse	+4	FR_A_RX
11	0xB		B			FR_B_RX
12	0xC	content error detected	A	level	+4	FR_A_RX
13	0xD		B			FR_B_RX
14	0xE	receive FIFO almost-full interrupt signals	A	value	n.a.	RX FIFO A Almost Full Interrupt

Table continues on the next page...

FR_STBSCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<b>Table 50-8. Strobe Signal Mapping (continued)</b>
<table>
<tr><th colspan="2">SEL</th><th rowspan="2">Description</th><th rowspan="2">Channel</th><th rowspan="2">Type</th><th rowspan="2">Offset (Given in PE clock cycles)</th><th rowspan="2">Reference</th></tr>
<tr><th>dec</th><th>hex</th></tr>
<tr><td>15</td><td>0xF</td><td></td><td>B</td><td></td><td></td><td>RX FIFO B Almost Full Interrupt</td></tr>
</table>
<table>
<tr><td>Write: Disabled Mode</td></tr>
</table>
</td></tr>
<tr><td>8–10
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>11
ENB</td><td>Strobe Signal Enable. The control bit is used to enable and to disable the strobe signal selected by STBSSEL.
0 Strobe signal is disabled and not assigned to any strobe port.
1 Strobe signal is enabled and assigned to the strobe port selected by STBPSEL.</td></tr>
<tr><td>12–13
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>14–15
STBPSEL</td><td>Strobe Port Select. This field selects the strobe port that the strobe signal selected by the SEL is assigned to. All strobe signals that are enabled and assigned to the same strobe port are combined with a binary OR operation.
00 assign selected signal to FR_DBG[0]
01 assign selected signal to FR_DBG[1]
10 assign selected signal to FR_DBG[2]
11 assign selected signal to FR_DBG[3]</td></tr>
</table>

50.6.6 Message Buffer Data Size Register (FR_MBDSR)

This register defines the size of the message buffer data section for the two message buffer segments in a number of two-byte entities.

The CC provides two independent segments for the individual message buffers. All individual message buffers within one segment have to have the same size for the message buffer data section. This size can be different for the two message buffer segments.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	MBSEG2DS							0	MBSEG1DS						
Write	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MBDSR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 MBSEG2DS	Message Buffer Segment 2 Data Size. The field defines the size of the message buffer data section in two-byte entities for message buffers within the second message buffer segment.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBSEG1DS	Message Buffer Segment 1 Data Size. The field defines the size of the message buffer data section in two-byte entities for message buffers within the first message buffer segment.

50.6.7 Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

This register is used to define the last individual message buffer that belongs to the first message buffer segment and the number of the last used individual message buffer.

Address: 0h base + Eh offset = Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		LAST_MB_SEG1						0		LAST_MB_UTIL					
Write	▨	▨							▨	▨						
Reset	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1

FR_MBSSUTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 LAST_MB_SEG1	Last Message Buffer In Segment 1. This field defines the message buffer number of the last individual message buffer that is assigned to the first message buffer segment. The individual message buffers in the first segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with n <= LAST_MB_SEG1. The first message buffer segment contains LAST_MB_SEG1+1 individual message buffers. **NOTE:** The first message buffer segment contains at least one individual message buffer. The individual message buffers in the second message buffer segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with LAST_MB_SEG1 < n < 64 256. **NOTE:** If LAST_MB_SEG1 = 63 255 all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.

Table continues on the next page...

FR_MBSSUTR field descriptions (continued)

Field	Description
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 LAST_MB_UTIL	Last Message Buffer Utilized. This field defines the message buffer number of last utilized individual message buffer. The message buffer search engine examines all individual message buffer with a message buffer number n <= LAST_MB_UTIL. **NOTE:** If LAST_MB_UTIL=LAST_MB_SEG1 all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.

50.6.8 PE DRAM Access Register (FR_PEDRAR)

The PEDRAR register is used to trigger write and read operations on the PE data memory (PE DRAM). These operations are used for memory error injection and memory error observation.

Each write access to this registers initiates a read or write operation on the PE DRAM. The access done status bit DAD is cleared after the write access and is set if the PE DRAM access has been finished.

In case of an PE DRAM write access, the data provided in FR_PEDRDR are written into the PE DRAM, read back from the PE DRAM and are stored into the FR_PEDRDR.

In case of an PE DRAM read access, the requested data are read from PE DRAM and stored into the FR_PEDRDR.

For a detailed description refer to Memory Content Fault Detection

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	INST				ADDR											DAD
Write																▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PEDRAR field descriptions

Field	Description
0–3 INST	PE DRAM Access Instruction. This field defines the operation to be executed on the PE DRAM. 0011 PE DRAM write: Write FR_PEDRDR[DATA] to PE DRAM address ADDR (16 bit) 0101 PE DRAM read: Read Data from PE DRAM address ADDR (16 bit) into FR_PEDRDR[DATA] other reserved
4–14 ADDR	PE DRAM Access Address. This field defines the address in the PE DRAM to be written to or read from.

Table continues on the next page...

FR_PEDRAR field descriptions (continued)

Field	Description
15 DAD	PE DRAM Access Done. This status bit is cleared when the application has written to this register and is set when the PE DRAM access has finished. 0 PE DRAM access running 1 PE DRAM access done

50.6.9 PE DRAM Data Register (FR_PEDRDR)

This register provides the data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

Address: 0h base + 12h offset = 12h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PEDRDR field descriptions

Field	Description
0–15 DATA	Data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

50.6.10 Protocol Operation Control Register (FR_POCR)

The application uses this register to issue

- protocol control commands
- external clock correction commands

Protocol control commands are issued by writing to the POCCMD field. For more information on protocol control commands, see Protocol Control Command Execution .

External clock correction commands are issued by writing to the EOC_AP and ERC_AP fields. For more information on external clock correction, refer to External Clock Synchronization .

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0			EOC_AP		ERC_AP		BSY_ WMC	0			POCCMD			
Write	WME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_POCR field descriptions

Field	Description
0 WME	Write Mode External Correction. This bit controls the write mode of the EOC_AP and ERC_AP fields. 0 Write to EOC_AP and ERC_AP fields on register write. 1 No write to EOC_AP and ERC_AP fields on register write.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–5 EOC_AP	External Offset Correction Application. This field is used to trigger the application of the external offset correction value defined in the Protocol Configuration Register 29 (FR_PCR29). 00 do not apply external offset correction value 01 reserved 10 subtract external offset correction value 11 add external offset correction value
6–7 ERC_AP	External Rate Correction Application. This field is used to trigger application of the external rate correction value defined in the Protocol Configuration Register 21 (FR_PCR21) 00 do not apply external rate correction value 01 reserved 10 subtract external rate correction value 11 add external rate correction value
8 BSY_WMC	**NOTE:** The name and function of this field varies depending on whether it is being read or written. BSY (Read-Only). Protocol Control Command Busy. This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR) , and will not change the value of the POCCMD field. WMC (Write-Only) Protocol Control Command Write. This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR) , and will not change the value of the POCCMD field. 0 (Write-Only) Write to POCCMD field on register write. 1 (Write-Only) Do not write to POCCMD field on register write. 0 (Read-Only) Command write idle, command accepted and ready to receive new protocol command. 1 (Read-Only) Command write busy, command not yet accepted, not ready to receive new protocol command.

Table continues on the next page...

FR_POCR field descriptions (continued)

Field	Description
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 POCCMD	Protocol Control Command. The application writes to this field to issue a protocol control command to the PE. The CC sends the protocol command to the PE immediately. While the transfer is running, the BSY bit is set. 0000 ALLOW_COLDSTART. Immediately activate capability of node to cold start cluster. 0001 ALL_SLOTS. Delayed transition to the all slots transmission mode. (Delayed means on completion of current communication cycle.) 0010 CONFIG. Immediately transition to the POC:config state. 0011 FREEZE. Immediately transition to the POC:halt state. 0100 READY, CONFIG_COMPLETE. Immediately transition to the POC:ready state. 0101 RUN. Immediately transition to the POC:startup start state. 0110 DEFAULT_CONFIG. Immediately transition to the POC:default config state. 0111 HALT. Delayed transition to the POC:halt state 1000 WAKEUP. Immediately initiate the wakeup procedure. 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

50.6.11 Global Interrupt Flag and Enable Register (FR_GIFER)

This register provides the means to control some of the interrupt request lines and provides the corresponding interrupt flags. The interrupt flags MIF, PRIF, CHIF, RBIF, and TBIF are the outcome of a binary OR of the related individual interrupt flags and interrupt enables . The generation scheme for these flags is depicted in Figure 50-33 . For more details on interrupt generation, see Interrupt Support . These flags are cleared automatically when all of the corresponding interrupt flags or interrupt enables in the related interrupt flag and enable registers are cleared by the application.

Address: 0h base + 16h offset = 16h

Bit	0	1	2	3	4	5	6	7
Read	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
Write				w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	MIE	PRIE	CHIE	WUPIE	FAFBIE	FAFAIE	RBIE	TBIE
Reset	0	0	0	0	0	0	0	0

FR_GIFER field descriptions

Field	Description
0 MIF	Module Interrupt Flag. This interrupt flag is set if at least one of the other interrupt flags in this register and the related interrupt enable bit are set. 0 No interrupt flag and related interrupt enable bit are set 1 At least one of the other interrupt flags in this register and the related interrupt bit are set.
1 PRIF	Protocol Interrupt Flag. This interrupt flag is set if at least one of the individual flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1) and the related interrupt enable bit are set. 0 No individual protocol interrupt flag and related interrupt enable bit are set. 1 At least one of the individual protocol interrupt flags and the related interrupt enable bit are set.
2 CHIF	CHI Interrupt Flag. This interrupt flag is set if at least one of the error flags in the CHI Error Flag Register (FR_CHIERFR) and the chi error interrupt enable bit FR_GIFER[CHIE] are set . 0 All CHI error flags are equal to 0 or the chi error interrupt is disabled. 1 At least one CHI error flag and the chi error interrupt enable are is set.
3 WUPIF	Wakeup Interrupt Flag. This interrupt flag is set when the CC has received a wakeup symbol on the FlexRay bus. The application can determine on which channel the wakeup symbol was received by reading the related wakeup flags WUB and WUA in the Protocol Status Register 3 (FR_PSR3).

Table continues on the next page...

FR_GIFER field descriptions (continued)

Field	Description
	0 No Wakeup symbol received on FlexRay bus 1 Wakeup symbol received on FlexRay bus
4 FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag. This interrupt flag is set when one of the following events occurs a) the current number of FIFO B entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO B, or b) the current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 no such event 1 FIFO B almost full event has occurred
5 FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag. This interrupt flag is set when one of the following events occurs a) the current number of FIFO A entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO A, or b) the current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 no such event 1 FIFO A almost full event has occurred
6 RBIF	Receive Message Buffer Interrupt Flag. This interrupt flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) both the interrupt flag MBIF and the interrupt enable bit MBIE in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are asserted. The application can not clear this interrupt flag directly, instead it is cleared by the CC when all of the interrupt flags MBIF of the individual receive message buffers are cleared by the application or if the application has cleared the related interrupt enables bit MBIE. 0 None of the individual receive message buffers has the MBIF and MBIE flag set. 1 At least one individual receive message buffer has the MBIF and MBIE flag set.
7 TBIF	Transmit Message Buffer Interrupt Flag. This flag is set if for at least one of the individual message buffers (FR_MBCCSRn[MTD] = 1) both the interrupt flag MBIF and the interrupt enable bit MBIE in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are equal to 1. The application can not clear this interrupt flag directly, instead, this interrupt flag is cleared by the CC when either all of the individual interrupt flags MBIF of the individual transmit message buffers are cleared by the application or the application has cleared the related interrupt enables bit MBIE. 0 None of the individual transmit message buffers has the MBIF and MBIE flag set. 1 At least one individual transmit message buffer has the MBIF and MBIE flag set.
8 MIE	Module Interrupt Enable. This bit controls if the Module Interrupt line is asserted when the MIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
9 PRIE	Protocol Interrupt Enable. This bit controls if the Protocol Interrupt line is asserted when the PRIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

Table continues on the next page...

FR_GIFER field descriptions (continued)

Field	Description
10 CHIE	CHI Interrupt Enable. This bit controls if the CHI Interrupt line is asserted when the CHIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
11 WUPIE	Wakeup Interrupt Enable. This bit controls if the Wakeup Interrupt line is asserted when the WUPIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
12 FAFBIE	Receive FIFO Channel B Almost Full Interrupt Enable. This bit controls if the RX FIFO B Almost Full Interrupt line is asserted when the FAFBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 FAFAIE	Receive FIFO Channel A Almost Full Interrupt Enable. This bit controls if the RX FIFO A Almost Full Interrupt line is asserted when the FAFAIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 RBIE	Receive Message Buffer Interrupt Enable. This bit controls if the Receive Message Buffer Interrupt line is asserted when the RBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 TBIE	Transmit Message Buffer Interrupt Enable. This bit controls if the Transmit Message Buffer Interrupt line is asserted when the TBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

50.6.12 Protocol Interrupt Flag Register 0 (FR_PIFR0)

The register holds one set of the protocol-related individual interrupt flags.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	FATL_IF	INTL_IF	ILCF_IF	CSA_IF	MRC_IF	MOC_IF	CCL_IF	MXS_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	MTX_IF	LTXB_IF	LTXA_IF	TBVB_IF	TBVA_IF	TI2_IF	TI1_IF	CYS_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_PIFR0 field descriptions

Field	Description
0 FATL_IF	Fatal Protocol Error Interrupt Flag. This flag is set when the protocol engine has detected a fatal protocol error. In this case, the protocol engine goes into the POC:halt state immediately. The fatal protocol errors are: 1) pLatestTx violation, as described in the MAC process of the FlexRay protocol 2) transmission across slot boundary violation, as described in the FSP process of the FlexRay protocol 0 No such event. 1 Fatal protocol error detected.
1 INTL_IF	Internal Protocol Error Interrupt Flag. This flag is set when the protocol engine has detected an internal protocol error. In this case, the protocol engine goes into the POC:halt state immediately. An internal protocol error occurs when the protocol engine has not finished a calculation and a new calculation is requested. This can be caused by a hardware error. 0 No such event. 1 Internal protocol error detected.
2 ILCF_IF	Illegal Protocol Configuration Interrupt Flag. This flag is set when the protocol engine has detected an illegal protocol configuration parameter setting. In this case, the protocol engine goes into the POC:halt state immediately. The protocol engine checks the listen_timeout value programmed into the Protocol Configuration Register 14 (FR_PCR14) and Protocol Configuration Register 15 (FR_PCR15) when the CONFIG_COMPLETE command was sent by the application via the Protocol Operation Control Register (FR_POCR). If the value of listen_timeout is equal to zero, the protocol configuration setting is considered as illegal. 0 No such event. 1 Illegal protocol configuration detected.
3 CSA_IF	Cold Start Abort Interrupt Flag. This flag is set when the configured number of allowed cold start attempts is reached and none of these attempts was successful. The number of allowed cold start attempts is configured by the coldstart_attempts field in the Protocol Configuration Register 3 (FR_PCR3). 0 No such event. 1 Cold start aborted and no more coldstart attempts allowed.
4 MRC_IF	Missing Rate Correction Interrupt Flag. This flag is set when an insufficient number of measurements is available for rate correction at the end of the communication cycle. 0 No such event 1 Insufficient number of measurements for rate correction detected
5 MOC_IF	Missing Offset Correction Interrupt Flag. This flag is set when an insufficient number of measurements is available for offset correction. This is related to the MISSING_TERM event in the CSP process for offset correction in the FlexRay protocol. 0 No such event. 1 Insufficient number of measurements for offset correction detected.
6 CCL_IF	Clock Correction Limit Reached Interrupt Flag. This flag is set when the internal calculated offset or rate calculation values have reached or exceeded its configured thresholds as given by the offset_coorection_out field in the Protocol Configuration Register 9 (FR_PCR9) and the rate_correction_out field in the Protocol Configuration Register 14 (FR_PCR14). 0 No such event. 1 Offset or rate correction limit reached.

Table continues on the next page...

FR_PIFR0 field descriptions (continued)

Field	Description
7 MXS_IF	Max Sync Frames Detected Interrupt Flag. This flag is set when the number of synchronization frames detected in the current communication cycle exceeds the value of the node_sync_max field in the Protocol Configuration Register 30 (FR_PCR30). **NOTE:** Only synchronization frames that have passed the synchronization frame acceptance and rejection filters are taken into account. 0 No such event. 1 More than node_sync_max sync frames detected.
8 MTX_IF	Media Access Test Symbol Received Interrupt Flag. This flag is set when the MTS symbol was received on channel A or channel B. 0 No such event. 1 MTS symbol received.
9 LTXB_IF	pLatestTx Violation on Channel B Interrupt Flag. This flag is set when the frame transmission on channel B in the dynamic segment exceeds the dynamic segment boundary. This is related to the pLatestTx violation, as described in the MAC process of the FlexRay protocol. 0 No such event. 1 pLatestTx violation occurred on channel B.
10 LTXA_IF	pLatestTx Violation on Channel A Interrupt Flag. This flag is set when the frame transmission on channel A in the dynamic segment exceeds the dynamic segment boundary. This is related to the pLatestTx violation as described in the MAC process of the FlexRay protocol. 0 No such event. 1 pLatestTx violation occurred on channel A.
11 TBVB_IF	Transmission across boundary on channel B Interrupt Flag. This flag is set when the frame transmission on channel B crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event. 1 Transmission across boundary violation occurred on channel B.
12 TBVA_IF	Transmission across boundary on channel A Interrupt Flag. This flag is set when the frame transmission on channel A crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event. 1 Transmission across boundary violation occurred on channel A.
13 TI2_IF	Timer 2 Expired Interrupt Flag. This flag is set whenever timer 2 expires. 0 No such event. 1 Timer 2 has reached its time limit.
14 TI1_IF	Timer 1 Expired Interrupt Flag. This flag is set whenever timer 1 expires. 0 No such event 1 Timer 1 has reached its time limit
15 CYS_IF	Cycle Start Interrupt Flag. This flag is set when a communication cycle starts. 0 No such event 1 Communication cycle started.

50.6.13 Protocol Interrupt Flag Register 1 (FR_PIFR1)

The register holds one set of the protocol-related individual interrupt flags.

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7
Read	EMC_IF	IPC_IF	PECF_IF	PSC_IF	SSI3_IF	SSI2_IF	SSI1_IF	SSI0_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		EVT_IF	ODT_IF	0			
Write			w1c	w1c				
Reset	0	0	0	0	0	0	0	0

FR_PIFR1 field descriptions

Field	Description
0 EMC_IF	Error Mode Changed Interrupt Flag. This flag is set when the value of the ERRMODE bit field in the Protocol Status Register 0 (FR_PSR0) is changed by the CC. 0 No such event. 1 ERRMODE field changed.
1 IPC_IF	Illegal Protocol Control Command Interrupt Flag. This flag is set when the PE tries to execute a protocol control command, which was issued via the POCCMD field of the Protocol Operation Control Register (FR_POCR), and detects that this protocol control command is not allowed in the current protocol state. In this case the command is not executed. For more details, see Protocol Control Command Execution . 0 No such event. 1 Illegal protocol control command detected.
2 PECF_IF	Protocol Engine Communication Failure Interrupt Flag. This flag is set if the CC has detected a communication failure between the PE and the CHI. 0 No such event. 1 Protocol Engine Communication Failure detected.
3 PSC_IF	Protocol State Changed Interrupt Flag. This flag is set when the protocol state in the PROTSTATE field in the Protocol Status Register 0 (FR_PSR0) has changed. 0 No such event. 1 Protocol state changed.
4 SSI3_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
5 SSI2_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented.

Table continues on the next page...

FR_PIFR1 field descriptions (continued)

Field	Description
	0 No such event. 1 The corresponding slot status counter has incremented.
6 SSI1_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
7 SSI0_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 EVT_IF	Even Cycle Table Written Interrupt Flag. This flag is set if the CC has written the sync frame measurement / ID tables into the FlexRay memory area for the even cycle. 0 No such event. 1 Sync frame measurement table written
11 ODT_IF	Odd Cycle Table Written Interrupt Flag. This flag is set if the CC has written the sync frame measurement / ID tables into the FlexRay memory area for the odd cycle. 0 No such event. 1 Sync frame measurement table written
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

50.6.14 Protocol Interrupt Enable Register 0 (FR_PIER0)

This register defines whether or not the individual interrupt flags defined in the Protocol Interrupt Flag Register 0 (FR_PIFR0) can generate a protocol interrupt request.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read Write	FATL_IE	INTL_IE	ILCF_IE	CSA_IE	MRC_IE	MOC_IE	CCL_IE	MXS_IE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	MTX_IE	LTXB_IE	LTXA_IE	TBVB_IE	TBVA_IE	TI2_IE	TI1_IE	CYS_IE
Reset	0	0	0	0	0	0	0	0

FR_PIER0 field descriptions

Field	Description
0 FATL_IE	Fatal Protocol Error Interrupt Enable. This bit controls FATL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 INTL_IE	Internal Protocol Error Interrupt Enable. This bit controls INTL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
2 ILCF_IE	Illegal Protocol Configuration Interrupt Enable. This bit controls ILCF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
3 CSA_IE	Cold Start Abort Interrupt Enable. This bit controls CSA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 MRC_IE	Missing Rate Correction Interrupt Enable. This bit controls MRC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 MOC_IE	Missing Offset Correction Interrupt Enable. This bit controls MOC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 CCL_IE	Clock Correction Limit Reached Interrupt Enable. This bit controls CCL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
7 MXS_IE	Max Sync Frames Detected Interrupt Enable. This bit controls MXS_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8 MTX_IE	Media Access Test Symbol Received Interrupt Enable. This bit controls MTX_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
9 LTXB_IE	pLatestTx Violation on Channel B Interrupt Enable. This bit controls LTXB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
10 LTXA_IE	pLatestTx Violation on Channel A Interrupt Enable. This bit controls LTXA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 TBVB_IE	Transmission across boundary on channel B Interrupt Enable. This bit controls TBVB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table continues on the next page...

FR_PIER0 field descriptions (continued)

Field	Description
12 TBVA_IE	Transmission across boundary on channel A Interrupt Enable. This bit controls TBVA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
13 TI2_IE	Timer 2 Expired Interrupt Enable. This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
14 TI1_IE	Timer 1 Expired Interrupt Enable. This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
15 CYS_IE	Cycle Start Interrupt Enable. This bit controls CYC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

50.6.15 Protocol Interrupt Enable Register 1 (FR_PIER1)

This register defines whether or not the individual interrupt flags defined in Protocol Interrupt Flag Register 1 (FR_PIFR1) can generate a protocol interrupt request.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read Write	EMC_IE	IPC_IE	PECF_IE	PSC_IE	SSI3_IE	SSI2_IE	SSI1_IE	SSI0_IE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		EVT_IE	ODT_IE	0			
Write	▨	▨			▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

FR_PIER1 field descriptions

Field	Description
0 EMC_IE	Error Mode Changed Interrupt Enable. This bit controls EMC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 IPC_IE	Illegal Protocol Control Command Interrupt Enable. This bit controls IPC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table continues on the next page...

FR_PIER1 field descriptions (continued)

Field	Description
2 PECF_IE	Protocol Engine Communication Failure Interrupt Enable. This bit controls PECF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
3 PSC_IE	Protocol State Changed Interrupt Enable. This bit controls PSC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 SSI3_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 SSI2_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 SSI1_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
7 SSI0_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 EVT_IE	Even Cycle Table Written Interrupt Enable. This bit controls EVT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 ODT_IE	Odd Cycle Table Written Interrupt Enable. This bit controls ODT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

50.6.16 CHI Error Flag Register (FR_CHIERFR)

This register holds the CHI related error flags. The interrupt generation for each of these error flags is controlled by the CHI interrupt enable bit CHIE in the Global Interrupt Flag and Enable Register (FR_GIFER).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7
Read	FRLB_EF	FRLA_EF	PCMI_EF	FOVB_EF	FOVA_EF	MBS_EF	MBU_EF	LCK_EF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0	SBCF_EF	FID_EF	DPL_EF	SPL_EF	NML_EF	NMF_EF	ILSA_EF
Write		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_CHIERFR field descriptions

Field	Description
0 FRLB_EF	Frame Lost Channel B Error Flag. This flag is set if a complete frame was received on channel B but could not be stored in the selected individual message buffer because this message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such event 1 Frame lost on channel B detected
1 FRLA_EF	Frame Lost Channel A Error Flag. This flag is set if a complete frame was received on channel A but could not be stored in the selected individual message buffer because this message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such error 1 Frame lost on channel A detected
2 PCMI_EF	Protocol Command Ignored Error Flag. This flag is set if the application has issued a POC command by writing to the POCCMD field in the Protocol Operation Control Register (FR_POCR) while the BSY flag is equal to 1. In this case the command is ignored by the CC and is lost. 0 No such error 1 POC command ignored
3 FOVB_EF	Receive FIFO Overrun Channel B Error Flag. This flag is set when an overrun of the FIFO for channel B occurred. This error occurs if a semantically valid frame was received on channel B and matches the all criteria to be appended to the FIFO for channel B but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected

Table continues on the next page...

FR_CHIERFR field descriptions (continued)

Field	Description
4 FOVA_EF	Receive FIFO Overrun Channel A Error Flag. This flag is set when an overrun of the FIFO for channel A occurred. This error occurs if a semantically valid frame was received on channel A and matches the all criteria to be appended to the FIFO for channel A but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
5 MBS_EF	Message Buffer Search Error Flag. This flag is set if at least one of the following events occurs: a) The message buffer search engine is still running while the next search must be started due to the FlexRay protocol timing. b) A message buffer index greater than 67 is detected in the FR_MBIDXR[MBIDX] field of an found message buffer or in one of the FR_RSBIR[RSBIDX] fields. Refer to Message Buffer Search Error for details. 0 No such event 1 Search engine active while search start appears or illegal message buffer index detected
6 MBU_EF	Message Buffer Utilization Error Flag. This flag is asserted if the application writes to a message buffer control field that is beyond the number of utilized message buffers programmed in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). If the application writes to a FR_MBCCSRn register with n > LAST_MB_UTIL, the CC ignores the write attempt and asserts the message buffer utilization error flag MBU_EF in the CHI Error Flag Register (FR_CHIERFR). 0 No such event 1 Non-utilized message buffer enabled
7 LCK_EF	Lock Error Flag. This flag is set if the application tries to lock a message buffer that is already locked by the CC due to internal operations. In that case, the CC does not grant the lock to the application. The application must issue the lock request again. 0 No such error 1 Lock error detected
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SBCF_EF	System Bus Communication Failure Error Flag. This flag is set if a system bus access was not finished within the required amount of time (see System Bus Access Timeout). 0 No such event 1 System bus access not finished in time
10 FID_EF	Frame ID Error Flag. This flag is set if the frame ID stored in the message buffer header area differs from the frame ID stored in the message buffer control register . 0 No such error occurred 1 Frame ID error occurred
11 DPL_EF	Dynamic Payload Length Error Flag. This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the dynamic segment is greater than the maximum payload length for the dynamic segment as it is configured in the corresponding protocol configuration register field max_payload_length_dynamic in the Protocol Configuration Register 24 (FR_PCR24). 0 No such error occurred 1 Dynamic payload length error occurred

Table continues on the next page...

FR_CHIERFR field descriptions (continued)

Field	Description
12 SPL_EF	Static Payload Length Error Flag. This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the static segment is different from the payload length for the static segment as it is configured in the corresponding protocol configuration register field payload_length_static in the Protocol Configuration Register 19 (FR_PCR19). 0 No such error occurred 1 Static payload length error occurred
13 NML_EF	Network Management Length Error Flag. This flag is set if the payload length written into the header structure of a receive message buffer assigned to the static segment is less than the configured length of the Network Management Vector as configured in the Network Management Vector Length Register (FR_NMVLR). In this case the received part of the Network Management Vector will be used to update the Network Management Vector. 0 No such error occurred 1 Network management length error occurred
14 NMF_EF	Network Management Frame Error Flag. This flag is set if a received message in the static segment with a Preamble Indicator flag PP asserted has its Null Frame indicator flag NF asserted as well. In this case, the Global Network Management Registers (see Network Management Vector Registers (FR_NMVR0-FR_NMVR5)) are not updated. 0 No such error occurred 1 Network management frame error occurred
15 ILSA_EF	Illegal System Bus Address Error Flag. This flag is set if the external system bus subsystem has detected an access to an illegal system bus address from the CC (see System Bus Illegal Address Access). 0 No such event 1 Illegal system bus address accessed

50.6.17 Message Buffer Interrupt Vector Register (FR_MBIVEC)

This register indicates the lowest numbered receive message buffer and the lowest numbered transmit message buffer that have their interrupt status flag MBIF and interrupt enable MBIE bits asserted. This means that message buffers with lower message buffer numbers have higher priority.

Address: 0h base + 22h offset = 22h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		TBIVEC						0		RBIVEC					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MBIVEC field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 TBIVEC	Transmit Buffer Interrupt Vector. This field provides the number of the lowest numbered enabled transmit message buffer that has its interrupt status flag MBIF and its interrupt enable bit MBIE set. If there is no transmit message buffer with the interrupt status flag MBIF and the interrupt enable MBIE bits asserted, the value in this field is set to 0.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 RBIVEC	Receive Buffer Interrupt Vector. This field provides the message buffer number of the lowest numbered receive message buffer which has its interrupt flag MBIF and its interrupt enable bit MBIE asserted. If there is no receive message buffer with the interrupt status flag MBIF and the interrupt enable MBIE bits asserted, the value in this field is set to 0.

50.6.18 Channel A Status Error Counter Register (FR_CASERCR)

This register provides the channel status error counter for channel A. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits vSS!SyntaxError, vSS!ContentError, vSS!BViolation, and vSS!TxConflict. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, see Slot Status Monitoring.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHAERSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CASERCR field descriptions

Field	Description
0–15 CHAERSCNT	Channel A Status Error Counter. This field provides the current value channel status error counter. The counter value is updated within the first macrotick of the following slot or segment.

50.6.19 Channel B Status Error Counter Register (FR_CBSERCR)

This register provides the channel status error counter for channel B. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits vSS!SyntaxError, vSS!ContentError, vSS!BViolation, and vSS!TxConflict. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring see Slot Status Monitoring.

Address: 0h base + 26h offset = 26h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHBERSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CBSERCR field descriptions

Field	Description
0–15 CHBERSCNT	Channel B Status Error Counter. This field provides the current channel status error count. The counter value is updated within the first macrotick of the following slot or segment.

50.6.20 Protocol Status Register 0 (FR_PSR0)

This register provides information about the current protocol status.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	ERRMODE		SLOTMODE		0	PROTSTATE			STARTUPSTATE				0	WAKEUPSTATUS		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PSR0 field descriptions

Field	Description
0–1 ERRMODE	Error Mode. protocol related variable: vPOC!ErrorMode. This field indicates the error mode of the protocol. 00 ACTIVE 01 PASSIVE

Table continues on the next page...

FR_PSR0 field descriptions (continued)

Field	Description
	10 COMM_HALT 11 reserved
2–3 SLOTMODE	Slot Mode. protocol related variable: vPOC!SlotMode. This field indicates the slot mode of the protocol. 00 SINGLE 01 ALL_PENDING 10 ALL 11 reserved
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 PROTSTATE	Protocol State. protocol related variable: vPOC!State. This field indicates the state of the protocol. 000 POC:default config 001 POC:config 010 POC:wakeup 011 POC:ready 100 POC:normal passive 101 POC:normal active 110 POC:halt 111 POC:startup
8–11 STARTUPSTATE	Startup State. protocol related variable: vPOC!StartupState. This field indicates the current sub-state of the startup procedure. 0000 reserved 0001 reserved 0010 POC:coldstart collision resolution 0011 POC:coldstart listen 0100 POC:integration consistency check 0101 POC:integration listen 0110 reserved 0111 POC:initialize schedule 1000 reserved 1001 reserved 1010 POC:coldstart consistency check 1011 reserved 1100 reserved 1101 POC:integration coldstart check 1110 POC:coldstart gap 1111 POC:coldstart join
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 WAKEUPSTATUS	Wakeup Status. protocol related variable: vPOC!WakeupStatus. This field provides the outcome of the execution of the wakeup mechanism. 000 UNDEFINED 001 RECEIVED_HEADER 010 RECEIVED_WUP 011 COLLISION_ HEADER

Table continues on the next page...

FR_PSR0 field descriptions (continued)

Field	Description
	100 COLLISION_WUP 101 COLLISION_UNKNOWN 110 TRANSMITTED 111 reserved

50.6.21 Protocol Status Register 1 (FR_PSR1)

Address: 0h base + 2Ah offset = 2Ah

Bit	0	1	2	3	4	5	6	7
Read	CSAA	CSP	0	REMCSAT				
Write	w1c							
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	CPN	HHR	FRZ	APTAC				
Write								
Reset	0	0	0	0	0	0	0	0

FR_PSR1 field descriptions

Field	Description
0 CSAA	Cold Start Attempt Aborted Flag. protocol related event: 'set coldstart abort indicator in CHI' This flag is set when the CC has aborted a cold start attempt. 0 No such event 1 Cold start attempt aborted
1 CSP	Leading Cold Start Path. This status bit is set when the CC has reached the POC:normal active state via the leading cold start path. This indicates that this node has started the network 0 No such event 1 POC:normal active reached from POC:startup state via leading cold start path
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 REMCSAT	Remaining Coldstart Attempts. protocol related variable: vRemainingColdstartAttempts This field provides the number of remaining cold start attempts that the CC will execute.
8 CPN	Leading Cold Start Path Noise. protocol related variable: vPOC!ColdstartNoise This status bit is set if the CC has reached the POC:normal active state via the leading cold start path under noise conditions. This indicates there was some activity on the FlexRay bus while the CC was starting up the cluster. 0 No such event 1 POC:normal active state was reached from POC:startup state via noisy leading cold start path
9 HHR	Host Halt Request Pending. protocol related variable: vPOC!CHIHaltRequest This status bit is set when CC receives the HALT command from the application via the Protocol Operation Control Register

Table continues on the next page...

FR_PSR1 field descriptions (continued)

Field	Description
	(FR_POCR). The CC clears this status bit after a hard reset condition or when the protocol is in the POC:default config state. 0 No such event 1 HALT command received
10 FRZ	Freeze Occurred. protocol related variable: vPOC!Freeze This status bit is set when the CC has reached the POC:halt state due to the host FREEZE command or due to an internal error condition requiring immediate halt. The CC clears this status bit after a hard reset condition or when the protocol is in the POC:default config state. 0 No such event 1 Immediate halt due to FREEZE or internal error condition
11–15 APTAC	Allow Passive to Active Counter. protocol related variable: vPOC!vAllowPassivetoActive This field provides the number of consecutive even/odd communication cycle pairs that have passed with valid rate and offset correction terms, but the protocol is still in the POC:normal passive state due to an application configured delay to enter POC:normal active state. This delay is defined by the allow_passive_to_active field in the Protocol Configuration Register 12 (FR_PCR12).

50.6.22 Protocol Status Register 2 (FR_PSR2)

This register provides a snapshot of status information about the Network Idle Time NIT, the Symbol Window and the clock synchronization. The NIT related status bits NBVB, NSEB, NBVA, and NSEA are updated by the CC after the end of the NIT and before the end of the first slot of the next communication cycle. The Symbol Window related status bits STCB, SBVB, SSEB, MTB, STCA, SBVA, SSEB, and MTA are updated by the CC after the end of the symbol window and before the end of the current communication cycle. If no symbol window is configured, the symbol window related status bits remain in their reset state. The clock synchronization related CLKCORRFAILCNT is updated by the CC after the end of the static segment and before the end of the current communication cycle.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7
Read	NBVB	NSEB	STCB	SBVB	SSEB	MTB	NBVA	NSEA
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	STCA	SBVA	SSEA	MTA	CKCORFCNT			
Write								
Reset	0	0	0	0	0	0	0	0

FR_PSR2 field descriptions

Field	Description
0 NBVB	NIT Boundary Violation on Channel B. protocol related variable: vSS!BViolation for NIT on channel B This status bit is set when there was some media activity on the FlexRay bus channel B at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
1 NSEB	NIT Syntax Error on Channel B. protocol related variable: vSS!SyntaxError for NIT on channel B This status bit is set when a syntax error was detected during NIT on channel B. 0 No such event 1 Syntax error detected
2 STCB	Symbol Window Transmit Conflict on Channel B. protocol related variable: vSS!TxConflict for symbol window on channel B This status bit is set if there was a transmission conflict during the symbol window on channel B. 0 No such event 1 Transmission conflict detected
3 SBVB	Symbol Window Boundary Violation on Channel B. protocol related variable: vSS!BViolation for symbol window on channel B This status bit is set if there was some media activity on the FlexRay bus channel B at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
4 SSEB	Symbol Window Syntax Error on Channel B. protocol related variable: vSS!SyntaxError for symbol window on channel B This status bit is set when a syntax error was detected during the symbol window on channel B. 0 No such event 1 Syntax error detected
5 MTB	Media Access Test Symbol MTS Received on Channel B. protocol related variable: vSS!ValidMTS for Symbol Window on channel B This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel B. 0 No such event 1 MTS symbol received
6 NBVA	NIT Boundary Violation on Channel A. protocol related variable: vSS!BViolation for NIT on channel A This status bit is set when there was some media activity on the FlexRay bus channel A at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
7 NSEA	NIT Syntax Error on Channel A. protocol related variable: vSS!SyntaxError for NIT on channel A This status bit is set when a syntax error was detected during NIT on channel A. 0 No such event 1 Syntax error detected

Table continues on the next page...

FR_PSR2 field descriptions (continued)

Field	Description
8 STCA	Symbol Window Transmit Conflict on Channel A. protocol related variable: vSS!TxConflict for symbol window on channel A This status bit is set if there was a transmission conflicts during the symbol window on channel A. 0 No such event 1 Transmission conflict detected
9 SBVA	Symbol Window Boundary Violation on Channel A. protocol related variable: vSS!BViolation for symbol window on channel A This status bit is set if there was some media activity on the FlexRay bus channel A at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
10 SSEA	Symbol Window Syntax Error on Channel A. protocol related variable: vSS!SyntaxError for symbol window on channel A This status bit is set when a syntax error was detected during the symbol window on channel A. 0 No such event 1 Syntax error detected
11 MTA	Media Access Test Symbol MTS Received on Channel A. protocol related variable: vSS!ValidMTS for symbol window on channel A This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel A. 1 MTS symbol received 0 No such event
12–15 CKCORFCNT	Clock Correction Failed Counter. protocol related variable: vClockCorrectionFailed This field provides the number of consecutive even/odd communication cycle pairs that have passed without clock synchronization having performed an offset or a rate correction due to lack of synchronization frames. It is not incremented when it has reached the configured value of either max_without_clock_correction_fatal or max_without_clock_correction_passive as defined in the Protocol Configuration Register 8 (FR_PCR8). The CC resets this counter on a hard reset condition, when the protocol enters the POC:normal active state, or when both the rate and offset correction terms have been calculated successfully.

50.6.23 Protocol Status Register 3 (FR_PSR3)

This register provides aggregated channel status information as an accrued status of channel activity for all communication slots, regardless of whether they are assigned for transmission or subscribed for reception. It provides accrued information for the symbol window, the NIT, and the wakeup status.

Address: 0h base + 2Eh offset = 2Eh

Bit	0	1	2	3	4	5	6	7
Read	0		WUB	ABVB	AACB	ACEB	ASEB	AVFB
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		WUA	ABVA	AACA	ACEA	ASEA	AVFA
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_PSR3 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 WUB	Wakeup Symbol Received on Channel B. This flag is set when a wakeup symbol was received on channel B. 0 No wakeup symbol received 1 Wakeup symbol received
3 ABVB	Aggregated Boundary Violation on Channel B. This flag is set when a boundary violation has been detected on channel B. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
4 AACB	Aggregated Additional Communication on Channel B. This flag is set when at least one valid frame was received on channel B in a slot that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
5 ACEB	Aggregated Content Error on Channel B. This flag is set when a content error has been detected on channel B. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected

Table continues on the next page...

FR_PSR3 field descriptions (continued)

Field	Description
6 ASEB	Aggregated Syntax Error on Channel B. This flag is set when a syntax error has been detected on channel B. Syntax errors are detected in the communication slots, the symbol window and the NIT. 0 No syntax error detected 1 Syntax errors detected
7 AVFB	Aggregated Valid Frame on Channel B. This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel B. 1 At least one syntactically valid frame received 0 No syntactically valid frames received
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 WUA	Wakeup Symbol Received on Channel A. This flag is set when a wakeup symbol was received on channel A. 0 No wakeup symbol received 1 Wakeup symbol received
11 ABVA	Aggregated Boundary Violation on Channel A. This flag is set when a boundary violation has been detected on channel A. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
12 AACA	Aggregated Additional Communication on Channel A. This flag is set when a valid frame was received in a slot on channel A that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
13 ACEA	Aggregated Content Error on Channel A. This flag is set when a content error has been detected on channel A. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
14 ASEA	Aggregated Syntax Error on Channel A. This flag is set when a syntax error has been detected on channel A. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
15 AVFA	Aggregated Valid Frame on Channel A. This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel A. 0 No syntactically valid frames received 1 At least one syntactically valid frame received

50.6.24 Macrotick Counter Register (FR_MTCTR)

This register provides the macrotick count of the current communication cycle.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		MTCT													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTCTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 MTCT	Macrotick Counter. protocol related variable: vMacrotick This field provides the macrotick count of the current communication cycle.

50.6.25 Cycle Counter Register (FR_CYCTR)

This register provides the number of the current communication cycle.

Address: 0h base + 32h offset = 32h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0										CYCCNT					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CYCTR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNT	Cycle Counter. protocol related variable: vCycleCounter This field provides the number of the current communication cycle. If the counter reaches the maximum value of 63, the counter wraps and starts from zero again.

50.6.26 Slot Counter Channel A Register (FR_SLTCTAR)

This register provides the number of the current slot in the current communication cycle for channel A.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					SLOTCNTA										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SLTCTAR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTCNTA	Slot Counter Value for Channel A. protocol related variable: vSlotCounter for channel A This field provides the number of the current slot in the current communication cycle.

50.6.27 Slot Counter Channel B Register (FR_SLTCTBR)

This register provides the number of the current slot in the current communication cycle for channel B.

Address: 0h base + 36h offset = 36h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					SLOTCNTB										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SLTCTBR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTCNTB	Slot Counter Value for Channel B. protocol related variable: vSlotCounter for channel B This field provides the number of the current slot in the current communication cycle.

50.6.28 Rate Correction Value Register (FR_RTCORVR)

This register provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT of each odd numbered communication cycle.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	RATECORR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RTCORVR field descriptions

Field	Description
0–15 RATECORR	Rate Correction Value. protocol related variable: vRateCorrection (before value limitation and external rate correction) This field provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external rate correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by rate_correction_out in the Protocol Configuration Register 13 (FR_PCR13), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **NOTE:** If the CC was not able to calculate a new rate correction term due to a lack of synchronization frames, the RATECORR value is not updated.

50.6.29 Offset Correction Value Register (FR_OFCORVR)

This register provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT.

Address: 0h base + 3Ah offset = 3Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	OFFSETCORR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_OFCORVR field descriptions

Field	Description
0–15 OFFSETCORR	Offset Correction Value. protocol related variable: vOffsetCorrection (before value limitation and external offset correction) This field provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external offset correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by offset_correction_out field in the Protocol Configuration Register 29 (FR_PCR29), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **NOTE:** If the CC was not able to calculate an new offset correction term due to a lack of synchronization frames, the OFFSETCORR value is not updated.

50.6.30 Combined Interrupt Flag Register (FR_CIFR)

This register provides five combined interrupt flags and a copy of three individual interrupt flags. The combined interrupt flags are the result of a binary OR of the values of other interrupt flags regardless of the state of the interrupt enable bits. The generation scheme for the combined interrupt flags is depicted in Figure 50-35 . The individual interrupt flags WUPIF, FAFBIF, and FAFAIF are copies of corresponding flags in the Global Interrupt Flag and Enable Register (FR_GIFER) and are provided here to simplify the application interrupt flag check. To clear the individual interrupt flags, the application must use the Global Interrupt Flag and Enable Register (FR_GIFER).

NOTE

The meanings of the combined status bits MIF, PRIF, CHIF, RBIF, and TBIF are different from those mentioned in the Global Interrupt Flag and Enable Register (FR_GIFER).

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
Write								
Reset	0	0	0	0	0	0	0	0

FR_CIFR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_CIFR field descriptions (continued)

Field	Description
8 MIF	Module Interrupt Flag. This flag is set if there is at least one interrupt source that has its interrupt flag asserted. 0 No interrupt source has its interrupt flag asserted 1 At least one interrupt source has its interrupt flag asserted
9 PRIF	Protocol Interrupt Flag. This flag is set if at least one of the individual protocol interrupt flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) or Protocol Interrupt Flag Register 1 (FR_PIFR1) is equal to 1. 0 All individual protocol interrupt flags are equal to 0 1 At least one of the individual protocol interrupt flags is equal to 1
10 CHIF	CHI Interrupt Flag. This flag is set if at least one of the individual CHI error flags in the CHI Error Flag Register (FR_CHIERFR) is equal to 1. 0 All CHI error flags are equal to 0 1 At least one CHI error flag is equal to 1
11 WUPIF	Wakeup Interrupt Flag. Provides the same value as FR_GIFER[WUPIF]
12 FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag. Provides the same value as FR_GIFER[FAFBIF]
13 FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag. Provides the same value as FR_GIFER[FAFAIF]
14 RBIF	Receive Message Buffer Interrupt Flag. This flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual receive message buffers has the MBIF flag asserted. 1 At least one individual receive message buffers has the MBIF flag asserted.
15 TBIF	Transmit Message Buffer Interrupt Flag. This flag is set if for at least one of the individual transmit message buffers (FR_MBCCSRn[MTD] = 1) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual transmit message buffers has the MBIF flag asserted. 1 At least one individual transmit message buffers has the MBIF flag asserted.

50.6.31 System Memory Access Time-Out Register (FR_SYMATOR)

Address: 0h base + 3Eh offset = 3Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								TIMEOUT							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

FR_SYMATOR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 TIMEOUT	System Memory Access Time-Out. This value defines when a system bus access timout is detected. For a detailed description see Configure System Memory Access Time-Out Register (FR_SYMATOR) and System Bus Access Timeout .

50.6.32 Sync Frame Counter Register (FR_SFCNTR)

This register provides the number of synchronization frames that are used for clock synchronization in the last even and in the last odd numbered communication cycle. This register is updated after the start of the NIT and before 10 MT after offset correction start.

NOTE

If the application has locked the even synchronization table at the end of the static segment of an even communication cycle, the CC will not update the fields SFEVB and SFEVA.

NOTE

If the application has locked the odd synchronization table at the end of the static segment of an odd communication cycle, the CC will not update the values SFODB and SFODA.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SFEVB				SFEVA				SFODB				SFODA			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFCNTR field descriptions

Field	Description
0–3 SFEVB	Sync Frames Channel B, even cycle. protocol related variable: size of (vsSyncIdListB for even cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
4–7 SFEVA	Sync Frames Channel A, even cycle. protocol related variable: size of (vsSyncIdListA for even cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
8–11 SFODB	Sync Frames Channel B, odd cycle. protocol related variable: size of (vsSyncIdListB for odd cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization .
12–15 SFODA	Sync Frames Channel A, odd cycle. protocol related variable: size of (vsSyncIdListA for odd cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization .

50.6.33 Sync Frame Table Offset Register (FR_SFTOR)

This register defines the FlexRay memory area related offset for sync frame tables. For more details, see Sync Frame ID and Sync Frame Deviation Tables .

Address: 0h base + 42h offset = 42h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SFT_OFFSET															0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFTOR field descriptions

Field	Description
0–14 SFT_OFFSET	Sync Frame Table Offset. The offset of the Sync Frame Tables in the FlexRay memory area. This offset is required to be 16-bit aligned. Thus STF_OFFSET[0] is always 0.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

50.6.34 Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

This register provides configuration, control, and status information related to the generation and access of the clock sync ID tables and clock sync measurement tables. For a detailed description, see Sync Frame ID and Sync Frame Deviation Tables .

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7
Read	0	0	CYCNUM					
Write	ELKT	OLKT						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	ELKS	OLKS	EVAL	OVAL	0	0	SDVEN	SIDEN
Write						OPT		
Reset	0	0	0	0	0	0	0	0

FR_SFTCCSR field descriptions

Field	Description
0 ELKT	Even Cycle Tables Lock/Unlock Trigger. This trigger bit is used to lock and unlock the even cycle tables. 0 No effect 1 Triggers lock/unlock of the even cycle tables.
1 OLKT	Odd Cycle Tables Lock/Unlock Trigger. This trigger bit is used to lock and unlock the odd cycle tables. 0 No effect 1 Triggers lock/unlock of the odd cycle tables.
2–7 CYCNUM	Cycle Number. This field provides the number of the cycle in which the currently locked table was recorded. If none or both tables are locked, this value is related to the even cycle table.
8 ELKS	Even Cycle Tables Lock Status. This status bit indicates whether the application has locked the even cycle tables. 0 Application has not locked the even cycle tables. 1 Application has locked the even cycle tables.
9 OLKS	Odd Cycle Tables Lock Status. This status bit indicates whether the application has locked the odd cycle tables. 0 Application has not locked the odd cycle tables. 1 Application has locked the odd cycle tables.
10 EVAL	Even Cycle Tables Valid. This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the even cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent).
11 OVAL	Odd Cycle Tables Valid. This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the odd cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent).
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 OPT	One Pair Trigger. This trigger bit controls whether the CC writes continuously or only one pair of Sync Frame Tables into the FlexRay memory area. If this trigger is set to 1 while SDVEN or SIDEN is set to 1, the CC writes only one pair of the enabled Sync Frame Tables corresponding to the next even-odd-cycle pair into the FlexRay memory area. In this case, the CC clears the SDVEN or SIDEN bits immediately. If this trigger is set to 0 while SDVEN or SIDEN is set to 1, the CC writes continuously the enabled Sync Frame Tables into the FlexRay memory area. 0 Write continuously pairs of enabled Sync Frame Tables into FlexRay memory area. 1 Write only one pair of enabled Sync Frame Tables into FlexRay memory area.
14 SDVEN	Sync Frame Deviation Table Enable. This bit controls the generation of the Sync Frame Deviation Tables. The application must set this bit to request the CC to write the Sync Frame Deviation Tables into the FlexRay memory area. **NOTE:** If SDVEN is set to 1, then SIDEN must also be set to 1.

Table continues on the next page...

FR_SFTCCSR field descriptions (continued)

Field	Description
	0 Do not write Sync Frame Deviation Tables 1 Write Sync Frame Deviation Tables into FlexRay memory area
15 SIDEN	Sync Frame ID Table Enable. This bit controls the generation of the Sync Frame ID Tables. The application must set this bit to 1 to request the CC to write the Sync Frame ID Tables into the FlexRay memory area. 0 Do not write Sync Frame ID Tables 1 Write Sync Frame ID Tables into FlexRay memory area

50.6.35 Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

This register defines the Sync Frame Rejection Filter ID. The application must update this register outside of the static segment. If the application updates this register in the static segment, it can appear that the CC accepts the sync frame in the current cycle.

Address: 0h base + 46h offset = 46h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						SYNFRID									
Write	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDRFR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 SYNFRID	Sync Frame Rejection ID. This field defines the frame ID of a frame that must not be used for clock synchronization. For details see Sync Frame Rejection Filtering .

50.6.36 Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

This register defines the sync frame acceptance filter value. For details on filtering, see Sync Frame Filtering .

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						FVAL									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDAFVR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FVAL	Filter Value. This field defines the value for the sync frame acceptance filtering.

50.6.37 Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

This register defines the sync frame acceptance filter mask. For details on filtering see Sync Frame Acceptance Filtering .

Address: 0h base + 4Ah offset = 4Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						FMSK									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDAFMR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FMSK	Filter Mask. This field defines the mask for the sync frame acceptance filtering.

50.6.38 Network Management Vector Register (FR_NMVR*n*)

Each of these six registers holds one part of the Network Management Vector. The length of the Network Management Vector is configured in the Network Management Vector Length Register (FR_NMVLR). If FR_NMVLR is programmed with a value that is less than 12 bytes, the remaining bytes of the Network Management Vector Registers (FR_NMVR0-FR_NMVR5), which are not used for the Network Management Vector accumulating, will remain 0.

The NMVR provides accrued information over all received NMVs in the last communication cycle. All NMVs received in one cycle are ORed into the NMVR. The NMVR is updated at the end of the communication cycle.

Address: 0h base + 4Ch offset + (2d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	NMVP															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_NMVR*n* field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–15
NMVP</td><td>Network Management Vector Part. The mapping between the Network Management Vector Registers (FR_NMVR0-FR_NMVR5) and the receive message buffer payload bytes in NMV[0:11] is depicted in the following table.
<b>Table 50-9. Mapping of NMVRn to the Received Payload Bytes NMVn</b>
<table>
<tr><th>NMVRn Register</th><th>NMVn Received Payload</th></tr>
<tr><td>FR_NMVR0[NMVP[15:8]]</td><td>NMV0</td></tr>
<tr><td>FR_NMVR0[NMVP[7:0]]</td><td>NMV1</td></tr>
<tr><td>FR_NMVR1[NMVP[15:8]]</td><td>NMV2</td></tr>
<tr><td>FR_NMVR1[NMVP[7:0]]</td><td>NMV3</td></tr>
<tr><td colspan="2">...</td></tr>
<tr><td>FR_NMVR5[NMVP[15:8]]</td><td>NMV10</td></tr>
<tr><td>FR_NMVR5[NMVP[7:0]]</td><td>NMV11</td></tr>
</table></td></tr>
</table>

50.6.39 Network Management Vector Length Register (FR_NMVLR)

This register defines the length of the network management vector in bytes.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0												NMVL			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_NMVLR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 NMVL	Network Management Vector Length. protocol related variable: gNetworkManagementVectorLength This field defines the length of the Network Management Vector in bytes. Legal values are between 0 and 12.

50.6.40 Timer Configuration and Control Register (FR_TICCR)

This register is used to configure and control the two timers T1 and T2. For timer details, see Timer Support . The Timer T1 is an absolute timer. The Timer T2 can be configured as an absolute or relative timer.

NOTE

Both timers are deactivated immediately when the protocol enters a state different from POC:normal active or POC:normal passive.

Address: 0h base + 5Ah offset = 5Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2_ CFG	T2_ REP	0	0	0	T2ST	0			T1_ REP	0	0	0	T1ST
Write						T2SP	T2TR							T1SP	T1TR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TICCR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_TICCR field descriptions (continued)

Field	Description
2 T2_CFG	Timer T2 Configuration. This bit configures the timebase mode of Timer T2. 0 T2 is absolute timer. 1 T2 is relative timer.
3 T2_REP	Timer T2 Repetitive Mode. This bit configures the repetition mode of Timer T2. 0 T2 is non repetitive 1 T2 is repetitive
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 T2SP	Timer T2 Stop. This trigger bit is used to stop timer T2. 0 no effect 1 stop timer T2
6 T2TR	Timer T2 Trigger. This trigger bit is used to start timer T2. 0 no effect 1 start timer T2
7 T2ST	Timer T2 State. This status bit provides the current state of timer T2. 0 timer T2 is idle 1 timer T2 is running
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 T1_REP	Timer T1 Repetitive Mode. This bit configures the repetition mode of timer T1. 0 T1 is non repetitive 1 T1 is repetitive
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 T1SP	Timer T1 Stop. This trigger bit is used to stop timer T1. 0 no effect 1 stop timer T1
14 T1TR	Timer T1 Trigger. This trigger bit is used to start timer T1. 0 no effect 1 start timer T1
15 T1ST	Timer T1 State. This status bit provides the current state of timer T1. 0 timer T1 is idle 1 timer T1 is running

50.6.41 Timer 1 Cycle Set Register (FR_TI1CYSR)

This register defines the cycle filter value and the cycle filter mask for timer T1. For a detailed description of timer T1, refer to Absolute Timer T1 .

NOTE

If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T1_CYC_VAL						0		T1_CYC_MSK					
Write	▨	▨							▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI1CYSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 T1_CYC_VAL	Timer T1 Cycle Filter Value. This field defines the cycle filter value for timer T1.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 T1_CYC_MSK	Timer T1 Cycle Filter Mask. This field defines the cycle filter mask for timer T1.

50.6.42 Timer 1 Macrotick Offset Register (FR_TI1MTOR)

This register holds the macrotick offset value for timer T1. For a detailed description of timer T1, refer to Absolute Timer T1 .

NOTE

If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

Address: 0h base + 5Eh offset = 5Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T1_MTOFFSET													
Write	▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI1MTOR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 T1_MTOFFSET	Timer 1 Macrotick Offset. This field defines the macrotick offset value for timer 1.

50.6.43 Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, .

NOTE

If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and timer T2 will expire according to the changed values.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2CYCVAL						0		T2CYCMSK					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR0_ABS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 T2CYCVAL	Timer T2 Cycle Filter Mask
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 T2CYCMSK	Timer T2 Cycle Filter Mask

50.6.44 Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Relative Timer T2.

NOTE

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	T2MTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR0_REL field descriptions

Field	Description
0–15 T2MTCNT	Timer T2 Macrotick High Word

50.6.45 Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Absolute Timer T2

NOTE

If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and the timer T2 will expire according to the changed values.

Address: 0h base + 62h offset = 62h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2MOFF													
Write	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR1_ABS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 T2MOFF	Timer T2 Macrotick Offset

50.6.46 Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Relative Timer T2.

NOTE

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

Address: 0h base + 62h offset = 62h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	T2MTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR1_REL field descriptions

Field	Description
0–15 T2MTCNT	Timer T2 Macrotick Low Word

50.6.47 Slot Status Selection Register (FR_SSSR)

This register is used to access the four internal non-memory mapped slot status selection registers FR_SSSR0 to FR_SSSR3. Each internal register selects a slot, or symbol window/NIT, whose status vector will be saved in the corresponding Slot Status Registers (FR_SSR0-FR_SSR7) according to Slot Status Selection Register (FR_SSSR) . For a detailed description of slot status monitoring, refer to Slot Status Monitoring .

Table 50-10. Mapping Between FR_SSSRn and FR_SSRn

Internal Slot Status Selection Register	Write the Slot Status of the Slot Selected by FR_SSSRn for each			
	Even Communication Cycle		Odd Communication Cycle	
	For Channel B to	For Channel A to	For Channel B to	For Channel A to
FR_SSSR0	FR_SSR0[15:8]	FR_SSR0[7:0]	FR_SSR1[15:8]	FR_SSR1[7:0]
FR_SSSR1	FR_SSR2[15:8]	FR_SSR2[7:0]	FR_SSR3[15:8]	FR_SSR3[7:0]

Table continues on the next page...

Table 50-10. Mapping Between FR_SSSRn and FR_SSRn (continued)

Internal Slot Status Selection Register	Write the Slot Status of the Slot Selected by FR_SSSRn for each			
	Even Communication Cycle		Odd Communication Cycle	
	For Channel B to	For Channel A to	For Channel B to	For Channel A to
FR_SSSR2	FR_SSR4[15:8]	FR_SSR4[7:0]	FR_SSR5[15:8]	FR_SSR5[7:0]
FR_SSSR3	FR_SSR6[15:8]	FR_SSR6[7:0]	FR_SSR7[15:8]	FR_SSR7[7:0]

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0	SEL		0	SLOTNUMBER										
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSSR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 SEL	Selector. This field selects one of the four internal slot status selection registers for access. 00 select FR_SSSR0. 01 select FR_SSSR1. 10 select FR_SSSR2. 11 select FR_SSSR3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTNUMBER	Slot Number. This field specifies the number of the slot whose status will be saved in the corresponding slot status registers. **NOTE:** If this value is set to 0, the related slot status register provides the status of the symbol window after the NIT start, and provides the status of the NIT after the cycle start.

50.6.48 Slot Status Counter Condition Register (FR_SSCCR)

This register is used to access and program the four internal non-memory mapped Slot Status Counter Condition Registers FR_SSCCR0 to FR_SSCCR3. Each of these four internal slot status counter condition registers defines the mode and the conditions for incrementing the counter in the corresponding Slot Status Counter Registers (FR_SSCR0-FR_SSCR3). The correspondence is given in Slot Status Counter Condition Register (FR_SSCCR) . For a detailed description of slot status counters, refer to Slot Status Counter Registers .

Table 50-11. Mapping between internal FR_SSCCRn and FR_SSCRn

Condition Register	Condition Defined for Register
FR_SSCCR0	FR_SSCR0
FR_SSCCR1	FR_SSCR1
FR_SSCCR2	FR_SSCR2
FR_SSCCR3	FR_SSCR3

Address: 0h base + 66h offset = 66h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0	SEL		0	CNTCFG		MCY	VFR	SYF	NUF	SUF	STATUSMASK			
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSCCR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 SEL	Selector. This field selects one of the four internal slot counter condition registers for access. 00 select FR_SSCCR0. 01 select FR_SSCCR1. 10 select FR_SSCCR2. 11 select FR_SSCCR3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–6 CNTCFG	Counter Configuration. These bit field controls the channel related incrementing of the slot status counter.

Table continues on the next page...

FR_SSCCR field descriptions (continued)

Field	Description
	00 increment by 1 if condition is fulfilled on channel A. 01 increment by 1 if condition is fulfilled on channel B. 10 increment by 1 if condition is fulfilled on at least one channel. 11 increment by 2 if condition is fulfilled on both channels channel; increment by 1 if condition is fulfilled on only one channel.
7 MCY	Multi Cycle Selection. This bit defines whether the slot status counter accumulates over multiple communication cycles or provides information for the previous communication cycle only. 0 The Slot Status Counter provides information for the previous communication cycle only. 1 The Slot Status Counter accumulates over multiple communication cycles.
8 VFR	Valid Frame Restriction. This bit is used to restrict the counter to received valid frames. 0 The counter is not restricted to valid frames only. 1 The counter is restricted to valid frames only.
9 SYF	Sync Frame Restriction. This bit is used to restrict the counter to received frames with the sync frame indicator bit set to 1. 0 The counter is not restricted with respect to the sync frame indicator bit. 1 The counter is restricted to frames with the sync frame indicator bit set to 1.
10 NUF	Null Frame Restriction. This bit is used to restrict the counter to received frames with the null frame indicator bit set to 0. 0 The counter is not restricted with respect to the null frame indicator bit. 1 The counter is restricted to frames with the null frame indicator bit set to 0.
11 SUF	Startup Frame Restriction. This bit is used to restrict the counter to received frames with the startup frame indicator bit set to 1. 0 The counter is not restricted with respect to the startup frame indicator bit. 1 The counter is restricted to received frames with the startup frame indicator bit set to 1.
12–15 STATUSMASK	Slot Status Mask. This bit field is used to enable the counter with respect to the four slot status error indicator bits. STATUSMASK[3] - This bit enables the counting for slots with the syntax error indicator bit set to 1. STATUSMASK[2] - This bit enables the counting for slots with the content error indicator bit set to 1. STATUSMASK[1] - This bit enables the counting for slots with the boundary violation indicator bit set to 1. STATUSMASK[0] - This bit enables the counting for slots with the transmission conflict indicator bit set to 1.

50.6.49 Slot Status Register (FR_SSR*n*)

Each Slot Status Register (SSR) holds the status vector of the slot specified in the corresponding internal slot status selection register, which can be programmed using the Slot Status Selection Register (FR_SSSR). Each register is updated after the end of the corresponding slot as shown in Figure 50-31 . The register bits are directly related to the protocol variables and described in more detail in Slot Status Monitoring .

Address: 0h base + 68h offset + (2d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSR*n* field descriptions

Field	Description
0 VFB	Valid Frame on Channel B. protocol related variable: vSS!ValidFrame channel B 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
1 SYB	Sync Frame Indicator Channel B. protocol related variable: vRF!Header!SyFIndicator channel B 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
2 NFB	Null Frame Indicator Channel B. protocol related variable: vRF!Header!NFIndicator channel B 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1
3 SUB	Startup Frame Indicator Channel B. protocol related variable: vRF!Header!SuFIndicator channel B 0 vRF!Header!SuFIndicator = 0 1 vRF!Header!SuFIndicator = 1
4 SEB	Syntax Error on Channel B. protocol related variable: vSS!SyntaxError channel B 0 vSS!SyntaxError = 0 1 vSS!SyntaxError = 1
5 CEB	Content Error on Channel B. protocol related variable: vSS!ContentError channel B 0 vSS!ContentError = 0 1 vSS!ContentError = 1
6 BVB	Boundary Violation on Channel B. protocol related variable: vSS!BViolation channel B 0 vSS!BViolation = 0 1 vSS!BViolation = 1

Table continues on the next page...

FR_SSR*n* field descriptions (continued)

Field	Description
7 TCB	Transmission Conflict on Channel B. protocol related variable: vSS!TxConflict channel B 0 vSS!TxConflict = 0 1 vSS!TxConflict = 1
8 VFA	Valid Frame on Channel A. protocol related variable: vSS!ValidFrame channel A 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
9 SYA	Sync Frame Indicator Channel A. protocol related variable: vRF!Header!SyFIndicator channel A 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
10 NFA	Null Frame Indicator Channel A. protocol related variable: vRF!Header!NFIndicator channel A 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1
11 SUA	Startup Frame Indicator Channel A. protocol related variable: vRF!Header!SuFIndicator channel A 0 vRF!Header!SuFIndicator = 0 1 vRF!Header!SuFIndicator = 1
12 SEA	Syntax Error on Channel A. protocol related variable: vSS!SyntaxError channel A 0 vSS!SyntaxError = 0 1 vSS!SyntaxError = 1
13 CEA	Content Error on Channel A. protocol related variable: vSS!ContentError channel A 0 vSS!ContentError = 0 1 vSS!ContentError = 1
14 BVA	Boundary Violation on Channel A. protocol related variable: vSS!BViolation channel A 0 vSS!BViolation = 0 1 vSS!BViolation = 1
15 TCA	Transmission Conflict on Channel A. protocol related variable: vSS!TxConflict channel A 0 vSS!TxConflict = 0 1 vSS!TxConflict = 1

50.6.50 Slot Status Counter Register (FR_SSCR*n*)

Each of these four registers provides the slot status counter value for the previous communication cycle(s) and is updated at the cycle start. The provided value depends on the control bits and fields in the related internal slot status counter condition register FR_SSCCRn, which can be programmed by using the Slot Status Counter Condition Register (FR_SSCCR). For more details, see Slot Status Counter Registers .

NOTE

If the counter has reached its maximum value 0xFFFF and is in the multicycle mode, i.e. FR_SSCCRn[MCY] = 1, the counter is not reset to 0x0000. The application can reset the counter by clearing the FR_SSCCRn[MCY] bit and waiting for the next cycle start, when the CC clears the counter. Subsequently, the counter can be set into the multicycle mode again.

NOTE

Address: 0h base + 78h offset + (2d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SLOTSTATUSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSCR*n* field descriptions

Field	Description
0–15 SLOTSTATUSCNT	Slot Status Counter. This field provides the current value of the Slot Status Counter.

50.6.51 MTS A Configuration Register (FR_MTSACFR)

This register controls the transmission of the Media Access Test Symbol MTS on channel A. For more details, see MTS Generation .

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MTE	0	CYCCNTMSK						0		CYCCNTVAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTSACFR field descriptions

Field	Description
0 MTE	Media Access Test Symbol Transmission Enable. This control bit is used to enable and disable the transmission of the Media Access Test Symbol in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_MTSACFR field descriptions (continued)

Field	Description
2–7 CYCCNTMSK	Cycle Counter Mask. This field provides the filter mask for the MTS cycle count filter.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNTVAL	Cycle Counter Value. This field provides the filter value for the MTS cycle count filter.

50.6.52 MTS B Configuration Register (FR_MTSBCFR)

This register controls the transmission of the Media Access Test Symbol MTS on channel B. For more details, see MTS Generation .

Address: 0h base + 82h offset = 82h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MTE	0	CYCCNTMSK						0		CYCCNTVAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTSBCFR field descriptions

Field	Description
0 MTE	Media Access Test Symbol Transmission Enable. This control bit is used to enable and disable the transmission of the Media Access Test Symbol in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 CYCCNTMSK	Cycle Counter Mask. This field provides the filter mask for the MTS cycle count filter.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNTVAL	Cycle Counter Value. This field provides the filter value for the MTS cycle count filter.

50.6.53 Receive Shadow Buffer Index Register (FR_RSBIR)

This register is used to provide and retrieve the indices of the message buffer header fields currently associated with the receive shadow buffers . For more details on the receive shadow buffer concept, refer to Receive Shadow Buffers Concept .

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7
Read Write	WMD	Reserved	SEL		Reserved			
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	Reserved	RSBIDX						
Reset	0	0	0	0	0	0	0	0

FR_RSBIR field descriptions

Field	Description
0 WMD	Write Mode. This bit controls the write mode for this register. 0 update SEL and RSBIDX field on register write 1 update only SEL field on register write
1 Reserved	This field is reserved.
2–3 SEL	Selector. This field is used to select the internal receive shadow buffer index register for access. 00 FR_RSBIR_A1. receive shadow buffer index register for channel A, segment 1 01 FR_RSBIR_A2. receive shadow buffer index register for channel A, segment 2 10 FR_RSBIR_B1. receive shadow buffer index register for channel B, segment 1 11 FR_RSBIR_B2. receive shadow buffer index register for channel B, segment 2
4–8 Reserved	This field is reserved.
9–15 RSBIDX	RSBIDXA1/RSBIDXA2/RSBIDXB1/RSBIDXB2- Receive Shadow Buffer Index Receive Shadow Buffer Index. This field contains the current index of the message buffer header field of the receive shadow message buffer selected by the SEL field. The CC uses this index to determine the physical location of the shadow buffer header field in the FlexRay memory area. The CC will update this field during receive operation.The application provides initial message buffer header index value in the configuration phase. CC: Updates the message buffer header index after successful reception. Application: Provides initial message buffer header index. Legal Values are 0 <= i <= 67 . Illegal values will be detected during the message buffer search.

50.6.54 Receive FIFO Watermark and Selection Register (FR_RFWMSR)

This register is used to

- select a receiver FIFO for subsequent programming access through the receiver FIFO configuration registers summarized in Receive FIFO Watermark and Selection Register (FR_RFWMSR) .
- to define the watermark for the selected FIFO.

Table 50-12. SEL Controlled Receiver FIFO Registers

Register
Receive FIFO Start Index Register (FR_RFSIR)
Receive FIFO Depth and Size Register (FR_RFDSR)
Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)
Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Address: 0h base + 86h offset = 86h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	WM								0							SEL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFWMSR field descriptions

Field	Description
0–7 WM	WMA/WMB - Watermark This field defines the watermark value for the selected FIFO. This value is used to control the generation of the almost full interrupt flags.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 SEL	Select. This control bit selects the receiver FIFO for subsequent programming. 0 Receiver FIFO for channel A selected 1 Receiver FIFO for channel B selected

50.6.55 Receive FIFO Start Index Register (FR_RFSIR)

This register defines the message buffer header index of the first message buffer of the selected FIFO.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						SIDX									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 SIDX	SIDXA/SIDXB - Start Index This field defines the number of the message buffer header field of the first message buffer of the selected FIFO. The CC uses the value of the SIDX field to determine the physical location of the receiver FIFO's first message buffer header field.

50.6.56 Receive FIFO Depth and Size Register (FR_RFDSR)

This register defines the structure of the selected FIFO, i.e. the number of entries and the size of each entry .

Address: 0h base + 8Ah offset = 8Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	FIFO_DEPTH								0	ENTRY_SIZE						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFDSR field descriptions

Field	Description
0–7 FIFO_DEPTH	FIFO_DEPTHA/FIFO_DEPTHB - FIFO Depth FIFO Depth. This field defines the depth of the selected FIFO, i.e. the number of entries. Note: If the FIFO_DEPTH is configured to 0, FR_RFFIDRFMR[FIDRFMSK] must be configured to 0 too, to ensure that no frames are received into the FIFO.

Table continues on the next page...

FR_RFDSR field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 ENTRY_SIZE	ENTRY_SIZEA/ENTRY_SIZEB - Entry Size This field defines the size of the frame data sections for the selected FIFO in 2 byte entities.

50.6.57 Receive FIFO A Read Index Register (FR_RFARIR)

This register provides the message buffer header index of the next available FIFO A entry that the application can read.

NOTE

If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						RDIDX									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFARIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 RDIDX	Read Index. This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

50.6.58 Receive FIFO B Read Index Register (FR_RFBRIR)

This register provides the message buffer header index of the next available FIFO B entry that the application can read.

NOTE

If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

Address: 0h base + 8Eh offset = 8Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						RDIDX									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFBRIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 RDIDX	Read Index. This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

50.6.59 Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

This register defines the filter value for the message ID acceptance filter of the selected FIFO. For details on message ID filtering see FIFO Filtering .

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MIDAFVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFMIDAFVR field descriptions

Field	Description
0–15 MIDAFVAL	MIDAFVALA/MIDAFVALB - Message ID Acceptance Filter Value Filter value for the message ID acceptance filter.

50.6.60 Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

This register defines the filter mask for the message ID acceptance filter of the selected FIFO. For details on message ID filtering see FIFO Filtering .

Address: 0h base + 92h offset = 92h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MIDAFMSK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFMIDAFMR field descriptions

Field	Description
0–15 MIDAFMSK	MIDAFMSKA/MIDAFMSKB - Message ID Acceptance Filter Mask Filter mask for the message ID acceptance filter.

50.6.61 Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

This register defines the filter value for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see FIFO Filtering .

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FIDRFVAL										
Write	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFIDRFVR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FIDRFVAL	FIDRFVALA/FIDRFVALB - Frame ID Rejection Filter Value Filter value for the frame ID rejection filter.

50.6.62 Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

This register defines the filter mask for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see FIFO Filtering .

Address: 0h base + 96h offset = 96h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FIDRFMSK										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFIDRFMR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FIDRFMSK	Frame ID Rejection Filter Mask. Filter mask for the frame ID rejection filter.

50.6.63 Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

This register provides access to the four internal frame ID range filter boundary registers of the selected FIFO. For details on frame ID range filter see FIFO Filtering .

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	IBD	SEL		0	SID										
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFRFCFR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL and IBD field only on write access.

Table continues on the next page...

FR_RFRFCFR field descriptions (continued)

Field	Description
1 IBD	Interval Boundary. This control bit selects the interval boundary to be programmed with the SID value. 0 program lower interval boundary 1 program upper interval boundary
2–3 SEL	Filter Selector. This control field selects the frame ID range filter to be accessed. 00 select frame ID range filter 0. 01 select frame ID range filter 1. 10 select frame ID range filter 2. 11 select frame ID range filter 3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SID	Slot ID Defines the IBD-selected frame ID boundary value for the SEL-selected range filter.

50.6.64 Receive FIFO Range Filter Control Register (FR_RFRFCTR)

This register is used to enable and disable each frame ID range filter and to define whether it is running as acceptance or rejection filter.

Address: 0h base + 9Ah offset = 9Ah

Bit	0	1	2	3	4	5	6	7
Read	0				F3MD	F2MD	F1MD	F0MD
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				F3EN	F2EN	F1EN	F0EN
Write								
Reset	0	0	0	0	0	0	0	0

FR_RFRFCTR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 F3MD	Range Filter 3 Mode. This control bit defines the filter mode of the frame ID range filter 3. 0 range filter 3 runs as acceptance filter 1 range filter 3 runs as rejection filter
5 F2MD	Range Filter 2 Mode. This control bit defines the filter mode of the frame ID range filter 2.

Table continues on the next page...

FR_RFRFCTR field descriptions (continued)

Field	Description
	0 range filter 2 runs as acceptance filter 1 range filter 2 runs as rejection filter
6 F1MD	Range Filter 1 Mode. This control bit defines the filter mode of the frame ID range filter 1. 0 range filter 1 runs as acceptance filter 1 range filter 1 runs as rejection filter
7 F0MD	Range Filter 0 Mode. This control bit defines the filter mode of the frame ID range filter 0. 0 range filter 0 runs as acceptance filter 1 range filter 0 runs as rejection filter
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 F3EN	Range Filter 3 Enable. This control bit is used to enable and disable the frame ID range filter 3. 0 range filter 3 disabled 1 range filter 3 enabled
13 F2EN	Range Filter 2 Enable. This control bit is used to enable and disable the frame ID range filter 2. 0 range filter 2 disabled 1 range filter 2 enabled
14 F1EN	Range Filter 1 Enable. This control bit is used to enable and disable the frame ID range filter 1. 0 range filter 1 disabled 1 range filter 1 enabled
15 F0EN	Range Filter 0 Enable. This control bit is used to enable and disable the frame ID range filter 0. 0 range filter 0 disabled 1 range filter 0 enabled

50.6.65 Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

This register provides the number of the last transmission slot in the dynamic segment for channel A. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					LDYNTXSLOTA										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_LDTXSLAR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 LDYNTXSLOTA	Last Dynamic Transmission Slot Channel A. protocol related variable: zLastDynTxSlot channel A Number of the last transmission slot in the dynamic segment for channel A. If no frame was transmitted during the dynamic segment on channel A, the value of this field is set to 0.

50.6.66 Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

This register provides the number of the last transmission slot in the dynamic segment for channel B. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Address: 0h base + 9Eh offset = 9Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					LDYNTXSLOTB										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_LDTXSLBR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 LDYNTXSLOTB	Last Dynamic Transmission Slot Channel B. protocol related variable: zLastDynTxSlot channel B Number of the last transmission slot in the dynamic segment for channel B. If no frame was transmitted during the dynamic segment on channel B the value of this field is set to 0.

50.6.67 Protocol Configuration Register 0 (FR_PCR0)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Table 50-13. Protocol Configuration Register (PCR0-30) Fields

Name	Description	Min	Max	Unit	FR_PCR
coldstart_attempts	gColdstartAttempts			number	3

Table continues on the next page...

Table 50-13. Protocol Configuration Register (PCR0-30) Fields (continued)

Name	Description	Min	Max	Unit	FR_PCR
action_point_offset	gdActionPointOffset - 1			MT	0
cas_rx_low_max	gdCASRxLowMax - 1			gdBit	4
dynamic_slot_idle_phase	gdDynamicSlotIdlePhase			minislot	28
minislot_action_point_offset	gdMinislotActionPointOffset - 1			MT	3
minislot_after_action_point	gdMinislot - gdMinislotActionPointOffset - 1			MT	2
static_slot_length	gdStaticSlot			MT	0
static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1			MT	13
symbol_window_exists	gdSymbolWindow!=0	0	1	bool	9
symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1			MT	6
tss_transmitter	gdTSSTransmitter			gdBit	5
wakeup_symbol_rx_idle	gdWakeupSymbolRxIdle			gdBit	5
wakeup_symbol_rx_low	gdWakeupSymbolRxLow			gdBit	3
wakeup_symbol_rx_window	gdWakeupSymbolRxWindow			gdBit	4
wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle			gdBit	8
wakeup_symbol_tx_low	gdWakeupSymbolTxLow			gdBit	5
noise_listen_timeout	(gListenNoise * pdListenTimeout) - 1			µT	16/17
macro_initial_offset_a	pMacroInitialOffset[A]			MT	6
macro_initial_offset_b	pMacroInitialOffset[B]			MT	16
macro_per_cycle	gMacroPerCycle			MT	10
macro_after_first_static_slot	gMacroPerCycle - gdStaticSlot			MT	1
macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart			MT	28
max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal			cyclepairs	8
max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive			cyclepairs	8
minislot_exists	gNumberOfMinislots!=0	0	1	bool	9
minislots_max	gNumberOfMinislots - 1			minislot	29
number_of_static_slots	gNumberOfStaticSlots			static slot	2
offset_correction_start	gOffsetCorrectionStart			MT	11
payload_length_static	gPayloadLengthStatic			2 bytes	19
max_payload_length_dynamic	pPayloadLengthDynMax			2 bytes	24
first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1			MT	13
allow_halt_due_to_clock	pAllowHaltDueToClock			bool	26
allow_passive_to_active	pAllowPassiveToActive			cyclepairs	12
cluster_drift_damping	pClusterDriftDamping			µT	24
comp_accepted_startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]			µT	22
comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]			µT	26
listen_timeout	pdListenTimeout - 1			µT	14/15

Table continues on the next page...

Table 50-13. Protocol Configuration Register (PCR0-30) Fields (continued)

Name	Description	Min	Max	Unit	FR_PCR
key_slot_id	pKeySlotId			number	18
key_slot_used_for_startup	pKeySlotUsedForStartup			bool	11
key_slot_used_for_sync	pKeySlotUsedForSync			bool	11
latest_tx	gNumberOfMinislots - pLatestTx			minislot	21
sync_node_max	gSyncNodeMax			number	30
micro_initial_offset_a	pMicroInitialOffset[A]			µT	20
micro_initial_offset_b	pMicroInitialOffset[B]			µT	20
micro_per_cycle	pMicroPerCycle			µT	22/23
micro_per_cycle_min	pMicroPerCycle - pdMaxDrift			µT	24/25
micro_per_cycle_max	pMicroPerCycle + pdMaxDrift			µT	26/27
micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)			µT	7
offset_correction_out	pOffsetCorrectionOut			µT	9
rate_correction_out	pRateCorrectionOut			µT	14
single_slot_enabled	pSingleSlotEnabled			bool	10
wakeup_channel	pWakeupChannel	see Protocol Configuration Register 0 (FR_PCR0)			10
wakeup_pattern	pWakeupPattern			number	18
decoding_correction_a	pDecodingCorrection + pDelayCompensation[A] + 2			µT	19
decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2			µT	7
key_slot_header_crc	header CRC for key slot	0x000	0x7FF	number	12
extern_offset_correction	pExternOffsetCorrection			µT	29
extern_rate_correction	pExternRateCorrection			µT	21

Table 50-14. Wakeup Channel Selection

wakeup_channel	Wakeup Channel
0	A
1	B

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	action_point_offset						static_slot_length									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR0 field descriptions

Field	Description
0–5 action_point_ offset	gdActionPointOffset - 1
6–15 static_slot_length	gdStaticSlot

50.6.68 Protocol Configuration Register 1 (FR_PCR1)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A2h offset = A2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		macro_after_first_static_slot													
Write	▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR1 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 macro_after_ first_static_slot	gMacroPerCycle - gdStaticSlot

50.6.69 Protocol Configuration Register 2 (FR_PCR2)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	minislot_after_action_point						number_of_static_slots									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR2 field descriptions

Field	Description
0–5 minislot_after_ action_point	gdMinislot - gdMinislotActionPointOffset - 1
6–15 number_of_ static_slots	gNumberOfStaticSlots

50.6.70 Protocol Configuration Register 3 (FR_PCR3)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A6h offset = A6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	wakeup_symbol_rx_low						minislot_action_point_offset					coldstart_attempts				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR3 field descriptions

Field	Description
0–5 wakeup_symbol_ rx_low	gdWakeupSymbolRxLow
6–10 minislot_action_ point_offset	gdMinislotActionPointOffset - 1
11–15 coldstart_ attempts	gColdstartAttempts

50.6.71 Protocol Configuration Register 4 (FR_PCR4)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	cas_rx_low_max							wakeup_symbol_rx_window								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR4 field descriptions

Field	Description
0–6 cas_rx_low_max	gdCASRxLowMax - 1
7–15 wakeup_symbol_ rx_window	gdWakeupSymbolRxWindow

50.6.72 Protocol Configuration Register 5 (FR_PCR5)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + AAh offset = AAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	tss_transmitter				wakeup_symbol_tx_low						wakeup_symbol_rx_idle					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR5 field descriptions

Field	Description
0–3 tss_transmitter	gdTSSTransmitter
4–9 wakeup_symbol_ tx_low	gdWakeupSymbolTxLow

Table continues on the next page...

FR_PCR5 field descriptions (continued)

Field	Description
10–15 wakeup_symbol_rx_idle	gdWakeupSymbolRxIdle

50.6.73 Protocol Configuration Register 6 (FR_PCR6)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	symbol_window_after_action_point								macro_initial_offset_a						
Write	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR6 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–8 symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1
9–15 macro_initial_offset_a	pMacroInitialOffset[A]

50.6.74 Protocol Configuration Register 7 (FR_PCR7)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + AEh offset = AEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	decoding_correction_b									micro_per_macro_nom_half						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR7 field descriptions

Field	Description
0–8 decoding_ correction_b	pDecodingCorrection + pDelayCompensation[B] + 2
9–15 micro_per_ macro_nom_half	round(pMicroPerMacroNom / 2)

50.6.75 Protocol Configuration Register 8 (FR_PCR8)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	max_without_clock_ correction_fatal				max_without_clock_ correction_passive				wakeup_symbol_tx_idle							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR8 field descriptions

Field	Description
0–3 max_without_ clock_correction_ fatal	gMaxWithoutClockCorrectionFatal

Table continues on the next page...

FR_PCR8 field descriptions (continued)

Field	Description
4–7 max_without_ clock_correction_ passive	gMaxWithoutClockCorrectionPassive
8–15 wakeup_symbol_ tx_idle	gdWakeupSymbolTxIdle

50.6.76 Protocol Configuration Register 9 (FR_PCR9)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B2h offset = B2h

Bit	0	1	2	3	4	5	6	7
Read Write	minislot_ exists	symbol_ window_ exists	offset_correction_out					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	offset_correction_out							
Reset	0	0	0	0	0	0	0	0

FR_PCR9 field descriptions

Field	Description
0 minislot_exists	gNumberOfMinislots!=0
1 symbol_window_ exists	gdSymbolWindow!=0
2–15 offset_ correction_out	pOffsetCorrectionOut

50.6.77 Protocol Configuration Register 10 (FR_PCR10)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7
Read Write	single_slot_ enabled	wakeup_ channel	macro_per_cycle					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	macro_per_cycle							
Reset	0	0	0	0	0	0	0	0

FR_PCR10 field descriptions

Field	Description
0 single_slot_ enabled	pSingleSlotEnabled
1 wakeup_channel	pWakeupChannel
2–15 macro_per_cycle	gMacroPerCycle

50.6.78 Protocol Configuration Register 11 (FR_PCR11)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B6h offset = B6h

Bit	0	1	2	3	4	5	6	7
Read Write	key_slot_used_for_startup	key_slot_used_for_sync	offset_correction_start					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	offset_correction_start							
Reset	0	0	0	0	0	0	0	0

FR_PCR11 field descriptions

Field	Description
0 key_slot_used_for_startup	pKeySlotUsedForStartup
1 key_slot_used_for_sync	pKeySlotUsedForSync
2–15 offset_correction_start	gOffsetCorrectionStart

50.6.79 Protocol Configuration Register 12 (FR_PCR12)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	allow_passive_to_active					key_slot_header_crc										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR12 field descriptions

Field	Description
0–4 allow_passive_to_active	pAllowPassiveToActive
5–15 key_slot_header_crc	header CRC for key slot

50.6.80 Protocol Configuration Register 13 (FR_PCR13)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BAh offset = BAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	first_minislot_action_point_offset						static_slot_after_action_point									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR13 field descriptions

Field	Description
0–5 first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1
6–15 static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1

50.6.81 Protocol Configuration Register 14 (FR_PCR14)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	rate_correction_out											listen_timeout				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR14 field descriptions

Field	Description
0–10 rate_correction_ out	pRateCorrectionOut
11–15 listen_timeout	pdListenTimeout - 1

50.6.82 Protocol Configuration Register 15 (FR_PCR15)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BEh offset = BEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	listen_timeout															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR15 field descriptions

Field	Description
0–15 listen_timeout	pdListenTimeout - 1

50.6.83 Protocol Configuration Register 16 (FR_PCR16)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	macro_initial_offset_b							noise_listen_timeout								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR16 field descriptions

Field	Description
0–6 macro_initial_ offset_b	pMacroInitialOffset[B]
7–15 noise_listen_ timeout	(gListenNoise * pdListenTimeout) - 1

50.6.84 Protocol Configuration Register 17 (FR_PCR17)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C2h offset = C2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	noise_listen_timeout															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR17 field descriptions

Field	Description
0–15 noise_listen_ timeout	(gListenNoise * pdListenTimeout) - 1

50.6.85 Protocol Configuration Register 18 (FR_PCR18)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	wakeup_pattern						key_slot_id									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR18 field descriptions

Field	Description
0–5 wakeup_pattern	pWakeupPattern
6–15 key_slot_id	pKeySlotId

50.6.86 Protocol Configuration Register 19 (FR_PCR19)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C6h offset = C6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	decoding_correction_a									payload_length_static						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR19 field descriptions

Field	Description
0–8 decoding_ correction_a	pDecodingCorrection + pDelayCompensation[A] + 2
9–15 payload_length_ static	gPayloadLengthStatic

50.6.87 Protocol Configuration Register 20 (FR_PCR20)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_initial_offset_b								micro_initial_offset_a							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR20 field descriptions

Field	Description
0–7 micro_initial_ offset_b	pMicroInitialOffset[A]
8–15 micro_initial_ offset_a	pMicroInitialOffset[A]

50.6.88 Protocol Configuration Register 21 (FR_PCR21)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CAh offset = CAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	extern_rate_ correction			latest_tx												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR21 field descriptions

Field	Description
0–2 extern_rate_ correction	pExternRateCorrection
3–15 latest_tx	gNumberOfMinislots - pLatestTx

50.6.89 Protocol Configuration Register 22 (FR_PCR22)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved	comp_accepted_startup_range_a						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	comp_accepted_startup_range_a				micro_per_cycle			
Reset	0	0	0	0	0	0	0	0

FR_PCR22 field descriptions

Field	Description
0 Reserved	Reserved bit, will not be changed. Application must not write any value different from the reset value. This field is reserved.
1–11 comp_accepted_ startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]
12–15 micro_per_cycle	pMicroPerCycle

50.6.90 Protocol Configuration Register 23 (FR_PCR23)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CEh offset = CEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR23 field descriptions

Field	Description
0–15 micro_per_cycle	pMicroPerCycle

50.6.91 Protocol Configuration Register 24 (FR_PCR24)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	cluster_drift_damping					max_payload_length_dynamic							micro_per_cycle_min			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR24 field descriptions

Field	Description
0–4 cluster_drift_ damping	pClusterDriftDamping
5–11 max_payload_ length_dynamic	pPayloadLengthDynMax
12–15 micro_per_cycle_ min	pMicroPerCycle - pdMaxDrift

50.6.92 Protocol Configuration Register 25 (FR_PCR25)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D2h offset = D2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle_min															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR25 field descriptions

Field	Description
0–15 micro_per_cycle_min	pMicroPerCycle - pdMaxDrift

50.6.93 Protocol Configuration Register 26 (FR_PCR26)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7
Read Write	allow_halt_due_to_clock	comp_accepted_startup_range_b						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	comp_accepted_startup_range_b				micro_per_cycle_max			
Reset	0	0	0	0	0	0	0	0

FR_PCR26 field descriptions

Field	Description
0 allow_halt_due_to_clock	pAllowHaltDueToClock
1–11 comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]
12–15 micro_per_cycle_max	pMicroPerCycle + pdMaxDrift

50.6.94 Protocol Configuration Register 27 (FR_PCR27)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D6h offset = D6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle_max															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR27 field descriptions

Field	Description
0–15 micro_per_cycle_max	pMicroPerCycle + pdMaxDrift

50.6.95 Protocol Configuration Register 28 (FR_PCR28)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	dynamic_slot_idle_phase		macro_after_offset_correction													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR28 field descriptions

Field	Description
0–1 dynamic_slot_idle_phase	gdDynamicSlotIdlePhase.
2–15 macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart

50.6.96 Protocol Configuration Register 29 (FR_PCR29)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + DAh offset = DAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	extern_offset_correction			minislots_max												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR29 field descriptions

Field	Description
0–2 extern_offset_correction	pExternOffsetCorrection
3–15 minislots_max	gNumberOfMinislots - 1

50.6.97 Protocol Configuration Register 30 (FR_PCR30)

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0												sync_node_max			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR30 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 sync_node_max	gSyncNodeMax

50.6.98 Protocol Event Output Enable Register (FR_PEOER)

This register defines whether or not the Event output ports are enabled based on the events such as Cycle start and Timer1 and Timer2 expiry (see Timer Support for details).

Address: 0h base + E2h offset = E2h

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0					TIM2_EE	TIM1_EE	CYC_EE
Write								
Reset	0	0	0	0	0	0	0	0

FR_PEOER field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 TIM2_EE	Timer 2 expired Event Output Enable- This bit controls the event on port "fr_evt_tim2" 0 Timer 2 expired event out disabled 1 Timer 2 expired event out enabled
14 TIM1_EE	Timer 1 expired Event Output Enable- This bit controls the event on port "fr_evt_tim1" 0 Timer 1 expired event out disabled 1 Timer 1 expired event out enabled
15 CYC_EE	Cycle Start Event Output Enable- This bit controls the event on port "fr_evt_cyc" 0 Cycle start event out disabled 1 Cycle start event out enabled

50.6.99 Receive FIFO Start Data Offset Register (FR_RFSDOR)

NOTE

Since all data fields of the FIFO are of equal length and are located at subsequent system memory addresses the content of the FR_RFSDOR corresponds to the start address of payload area of the selected FIFO.

Address: 0h base + E6h offset = E6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSDOR field descriptions

Field	Description
0–15 SDO	SDOA/SDOB - Start Data Field Offset Start Data Field Offset. This field defines the data field offset of the header field of the first message buffer of the selected FIFO. The CC uses the value of the SDO field to determine the physical location of the receiver FIFO's first message buffer header field. For configuration constraints see Configure Data Field Offsets .

50.6.100 Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSYMBADHR field descriptions

Field	Description
0–15 SMBA	System Memory Base Address. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.

50.6.101 Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Address: 0h base + EAh offset = EAh

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>Read</td><td colspan="12" rowspan="2">SMBA</td><td colspan="4">0</td></tr>
<tr><td>Write</td><td colspan="4"></td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

FR_RFSYMBADLR field descriptions

Field	Description
0–11 SMBA	System Memory Base Address. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

50.6.102 Receive FIFO Periodic Timer Register (FR_RFPTR)

This register holds periodic timer duration for the periodic FIFO timer. The periodic timer applies to both FIFOs (see FIFO Periodic Timer).

Address: 0h base + ECh offset = ECh

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>Read</td><td colspan="2">0</td><td colspan="14" rowspan="2">PTD</td></tr>
<tr><td>Write</td><td colspan="2"></td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

FR_RFPTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 PTD	Periodic Timer Duration. This value defines the periodic timer duration in terms of macroticks. 0000 timer stays expired 3FFF timer never expires other timer expires after specified number of macroticks, expires and is restarted at each cycle start

50.6.103 Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

This register provides the current fill level of the two receiver FIFOs and is used to pop a number of entries from the FIFOs

If the pop count value PCA/PCB is greater than the current FIFO fill level FLB/FLA, than the FIFO is empty after the update. No notification is given that not the required number of entries was removed.

.

Address: 0h base + EEh offset = EEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	FLB_or_PCB								FLA_or_PCA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFLPCR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–7
FLB_or_PCB</td><td>

NOTE: The name and function of the fields in this register vary depending on whether they are being read or written. See the following table for details.

<table>
<tr><th>Operation</th><th>Sub-Field</th><th>Description</th></tr>
<tr><td>Read</td><td>0-7 FLB</td><td>Fill Level FIFO B. This field provides the current number of entries in the FIFO B.</td></tr>
<tr><td></td><td>-</td><td>-</td></tr>
<tr><td>Write</td><td>0-7 PCB</td><td>Pop Count FIFO B. This field defines the number of entries to be removed from FIFO B.</td></tr>
<tr><td></td><td>-</td><td>-</td></tr>
</table>

</td></tr>
<tr><td>8–15
FLA_or_PCA</td><td></td></tr>
</table>

Table continues on the next page...

FR_RFFLPCR field descriptions (continued)

Field	Description		
	Operation	**Sub-Field**	**Description**
	Read	-	-
		8-15 FLA	Fill Level FIFO A- This field provides the current number of entries in the FIFO A.
	Write	-	-
		8-15 PCA	Pop Count FIFO A - This field defines the number of entries to be removed from FIFO A.

50.6.104 ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

This register provides the means to control the ECC related interrupt request lines and provides the corresponding interrupt flags. The interrupt flags are cleared by writing 1, which resets the corresponding report registers. For a detailed description see Memory Error Reporting .

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7
Read	LRNE_OF	LRCE_OF	DRNE_OF	DRCE_OF	LRNE_IF	LRCE_IF	DRNE_IF	DRCE_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				LRNE_IE	LRCE_IE	DRNE_IE	DRCE_IE
Write								
Reset	0	0	0	0	0	0	0	0

FR_EEIFER field descriptions

Field	Description
0 LRNE_OF	LRAM Non-Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected but not corrected on CHI LRAM and interrupt flag LRNE_IF is already 1. b) memory errors are detected but not corrected on at least two banks of CHI LRAM 0 no such event 1 Non-Corrected Error overflow detected on CHI LRAM
1 LRCE_OF	LRAM Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears

Table continues on the next page...

FR_EEIFER field descriptions (continued)

Field	Description
	a) memory errors are detected and corrected on CHI LRAM and interrupt flag LRCE_IF is already 1. b) memory errors are detected and corrected on at least two banks of CHI LRAM **NOTE:** Error Correction not implemented on CHI LRAM, flag will never be asserted. 0 no such event 1 Corrected Error overflow detected on CHI LRAM
2 DRNE_OF	DRAM Non-Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected but not corrected on PE DRAM and interrupt flag DRNE_IF is already 1. b) memory errors are detected but not corrected on at least two banks of the PE DRAM 0 no such event 1 Non-Corrected Error overflow detected on PE DRAM
3 DRCE_OF	DRAM Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected and corrected on PE DRAM and interrupt flag DRCE_IF is already 1. b) memory errors are detected and corrected on at least two banks of PE DRAM 0 no such event 1 Corrected Error overflow detected on PE DRAM
4 LRNE_IF	LRAM Non-Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected but not corrected on the CHI LRAM. 0 no such event 1 Non-Corrected Error detected on CHI LRAM
5 LRCE_IF	LRAM Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected and corrected on the CHI LRAM. **NOTE:** Error Correction not implemented on CHI LRAM, flag will never be asserted. 0 no such event 1 Corrected Error detected on CHI LRAM
6 DRNE_IF	DRAM Non-Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected but not corrected on PE DRAM. 0 no such event 1 Non-Corrected Error detected on PE DRAM
7 DRCE_IF	DRAM Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected and corrected on PE DRAM. 0 no such event 1 Corrected Error detected on PE DRAM
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 LRNE_IE	LRAM Non-Corrected Error Interrupt Enable. This flag controls if the LRAM Non-Corrected Error Interrupt line is asserted when the LRNE_IF flag is set.

Table continues on the next page...

FR_EEIFER field descriptions (continued)

Field	Description
	0 Disable interrupt line 1 Enable interrupt line
13 LRCE_IE	LRAM Corrected Error Interrupt Enable. This flag controls if the LRAM Corrected Error Interrupt line is asserted when the LRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 DRNE_IE	DRAM Non-Corrected Error Interrupt Enable. This flag controls if the DRAM Non-Corrected Error Interrupt line is asserted when the DRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 DRCE_IE	DRAM Corrected Error Interrupt Enable. This flag controls if the DRAM Corrected Error Interrupt line is asserted when the DRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line

50.6.105 ECC Error Report and Injection Control Register (FR_EERICR)

This register configures the error injection and error reporting and provides the selector for the content of the report registers.

Address: 0h base + F2h offset = F2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	BSY	0					ERS		0			ERM	0		EIM	EIE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERICR field descriptions

Field	Description
0 BSY	Register Update Busy- This field indicates the current state of the ECC configuration update and controls the register write access condition IDL specified in " Register Write Access 0 ECC configuration is idle 1 ECC configuration is running
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 ERS	Error Report Select. This field selects the content of the ECC Error reporting registers. 00 show PE DRAM non-corrected error information

Table continues on the next page...

FR_EERICR field descriptions (continued)

Field	Description
	01 show PE DRAM corrected error information 10 show CHI LRAM non-corrected error information 11 show CHI LRAM corrected error information
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ERM	Error Report Mode. This bit configures the type of data written into the internal error report registers on the detection of a memory error. 0 store data and code as delivered by ecc decoding logic. 1 store data and code as read from the memory.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 EIM	Error Injection Mode. This bit configures the ECC error injection mode. 0 use FR_EEIDR[DATA] and FR_EEICR[CODE] as XOR distortion pattern for error injection. 1 use FR_EEIDR[DATA] and FR_EEICR[CODE] as write value for error injection.
15 EIE	Error Injection Enable. This bit configures the ECC error injection on the memories. 0 Error injection disabled 1 Error injection enabled

50.6.106 ECC Error Report Address Register (FR_EERAR)

This register provides the memory identifier, bank, and address for which the memory error is reported.

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MID	BANK			ADDR											
Write																
Reset	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERAR field descriptions

Field	Description
0 MID	Memory Identifier. This flag provides the memory instance for which the memory error is reported. 0 PE DRAM 1 CHI LRAM
1–3 BANK	Memory Bank. This field provides the BANK for which the memory error is reported. 111 reset value, indicates no error found after reset. For MID=0:

Table continues on the next page...

FR_EERAR field descriptions (continued)

Field	Description
	000 PE DRAM [7:0] 001 PE DRAM [15:8] others - not used For MID=1: Refer to the following table for the assignment of the LRAM banks.

Table 50-15. LRAM Bank Value for MID = 1

BANK	Register		
000	FR_MBCCFR(2n)	FR_MBDOR(6n)	FR_LEETR0
001	FR_MBFIDR(2n)	FR_MBDOR(6n + 1)	FR_LEETR1
010	FR_MBIDXR(2n)	FR_MBDOR(6n + 2)	FR_LEETR2
011	FR_MBCCFR(2n+1)	FR_MBDOR(6n + 3)	FR_LEETR3
100	FR_MBFIDR(2n+1)	FR_MBDOR(6n + 4)	FR_LEETR4
101	FR_MBIDXR(2n+1)	FR_MBDOR(6n + 5)	FR_LEETR5
110	Not Used	Not Used	Not Used
111			

Field	Description
4–15 ADDR	Memory Address. This field provides the address of the failing memory location.

50.6.107 ECC Error Report Data Register (FR_EERDR)

This register provides the data related information of the reported memory read access. The assignment of the bits depends on the selected memory and memory bank as shown in ECC Error Report Data Register (FR_EERDR) .

Table 50-16. Valid Bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE DRAM	0	▨	▨	▨	▨	▨	▨	▨	▨	PE DRAM[7:0]							
PE DRAM	1	▨	▨	▨	▨	▨	▨	▨	▨	PE DRAM[15:8]							
CHI LRAM	0	FR_MBCCFR(2n)															
CHI LRAM	0	FR_MBDOR(6n)															
CHI LRAM	0	FR_LEETR0															
CHI LRAM	1	▨	▨	▨	▨	▨	FR_MBFIDR(2n)[FID]										
CHI LRAM	1	FR_MBDOR(6n+1)															
CHI LRAM	1	FR_LEETR1															
CHI LRAM	2	▨	▨	▨	▨	▨	▨	▨	▨	▨	FR_MBIDXR(2n)[MBIDX]						
CHI LRAM	3	FR_MBCCFR(2n+1)															
CHI LRAM	3	FR_MBDOR(6n+3)															

Table continues on the next page...

Table 50-16. Valid Bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field (continued)

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHI LRAM	3	FR_LEETR3															
CHI LRAM	4						FR_MBFIDR(2n+1)[FID]										
CHI LRAM	4	FR_MBDOR(6n+4)															
CHI LRAM	4	FR_LEETR4															
CHI LRAM	5										FR_MBIDXR(2n+1)[MBIDX]						
CHI LRAM	5	FR_MBDOR(6n+5)															
CHI LRAM	5	FR_LEETR5															

Address: 0h base + F6h offset = F6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DATA															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERDR field descriptions

Field	Description
0–15 DATA	Data. The content of this field depends on the report mode selected by FR_EERICR[ERM] ERM=0: Ecc Data, shows data as generated by the ecc decoding logic. ERM=1: Memory Data, shows data as read from the memory.

50.6.108 ECC Error Report Code Register (FR_EERCR)

This register provides the ECC related information of the reported memory read access.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0											CODE				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERCR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_EERCR field descriptions (continued)

Field	Description
11–15 CODE	Code. The content of this field depends on the report mode selected by FR_EERICR[ERM] ERM=0: Syndrome. Shows the ecc syndrome generated by the ecc decoding logic. The coding of the PE DRAM syndrome is shown in PE DRAM Syndrome . The coding of the CHI LRAM syndrome is shown in CHI LRAM Syndrome . ERM=1: Checkbits. Shows the ecc checkbits read from the memory.

50.6.109 ECC Error Injection Address Register (FR_EEIAR)

This register defines the memory module, bank, and address where the ECC error has to be injected.

Address: 0h base + FAh offset = FAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MID	BANK			ADDR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEIAR field descriptions

Field	Description
0 MID	Memory Identifier. This flag defines the memory instance for ECC error injection. 0 PE DRAM 1 CHI LRAM
1–3 BANK	Memory Bank. This field defines the memory bank for ECC error injection. For MID=0: 000 BANK0: PE DRAM [7:0] 001 BANK1: PE DRAM [15:8] others reserved For MID=1: Refer to ECC Error Report Address Register (FR_EERAR) for the assignment of the LRAM banks.
4–15 ADDR	Memory Address. This flag defines the memory address for ECC error injection.

50.6.110 ECC Error Injection Data Register (FR_EEIDR)

This register defines the data distortion pattern for the error injection write. The number of valid bits depends on the selected memory and memory bank as shown in ECC Error Report Data Register (FR_EERDR) .

NOTE

The effect of the error injected depends from the LRAM content at the address accessed and from the module internal usage of the data. Refer to Memory Error Response for details.

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEIDR field descriptions

Field	Description
0–15 DATA	Data. The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM=0: This field defines the XOR distortion pattern for the data written into the memory. EIM=1: This field defines the data to be written into the memory.

50.6.111 ECC Error Injection Code Register (FR_EEICR)

This register defines the ECC code distortion pattern for the error injection write.

Address: 0h base + FEh offset = FEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0											CODE				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEICR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CODE	Code. The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM=0: This field defines the XOR distortion pattern for the ecc checkbits written into the memory. EIM=1: This field defines the ecc checkbits written into the memory.

50.6.112 Message Buffer Configuration, Control, Status Register (FR_MBCCSR*n*)

The content of these registers is composed of message buffer configuration data, message buffer control data, message buffer status information, and message buffer interrupt flags. A detailed description of all flags can be found in Individual Message Buffer Functional Description.

If the application writes 1 to the EDT bit, no write access to the other register bits is performed.

If the application writes 0 to the EDT bit and 1 to the LCKT bit, no write access to the other bits is performed.

Address: 0h base + 800h offset + (8d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read	0			MTD	CMT		0	MBIE
Write						EDT	LCKT	
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			DUP	DVAL	EDS	LCKS	MBIF
Write								w1c
Reset	0	0	0	0	0	0	0	0

FR_MBCCSR*n* field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 MTD	Message Buffer Transfer Direction. This bit configures the transfer direction of a message buffer. 0 Receive message buffer 1 Transmit message buffer
4 CMT	Commit for Transmission. This bit indicates if the transmit message buffer data are ready for transmission. **NOTE:** This bit is read/write but may be modified by hardware other than by a reset. 0 Message buffer data not ready for transmission 1 Message buffer data ready for transmission
5 EDT	Enable/Disable Trigger. If the application writes 1 to this bit, a message buffer enable or disable is triggered, depending on the current value of the EDS status bit. 0 No effect 1 Message buffer enable or disable is triggered
6 LCKT	Lock/Unlock Trigger. If the application writes 1 to this bit and writes 0 to the EDT bit, a message buffer lock or unlock is triggered, depending on the current value of the LCKS status bit.

Table continues on the next page...

FR_MBCCSR*n* field descriptions (continued)

Field	Description
	0 No effect 1 Message buffer lock or unlock is triggered
7 MBIE	Message Buffer Interrupt Enable. This control bit defines whether the message buffer will generate an interrupt request when its MBIF flag is set. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 DUP	Data Updated. This status bit indicates whether the frame header in the message buffer header field and the data in the message buffer data field were updated after a frame reception. 0 Frame Header and Message buffer data field not updated 1 Frame Header and Message buffer data field updated
12 DVAL	Data Valid. For receive message buffers this status bit indicates whether the message buffer data field contains valid frame data. For transmit message buffers the status bit indicates if a message is transferred again due to the state transmission mode of the message buffer. 0 receive message buffer contains no valid frame data / message is transmitted for the first time 1 receive message buffer contains valid frame data / message will be transferred again
13 EDS	Enable/Disable Status. This status bit indicates whether the message buffer is enabled or disabled. 0 Message buffer is disabled. 1 Message buffer is enabled.
14 LCKS	Lock Status. This status bit indicates the current lock status of the message buffer. 0 Message buffer is not locked by the application. 1 Message buffer is locked by the application.
15 MBIF	Message Buffer Interrupt Flag. This flag is set when the slot status field of the message buffer was updated after frame transmission or reception, or when a transmit message buffer was just enabled by the application. 0 No such event 1 Slot status field updated or transmit message buffer just enabled

50.6.113 Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

This register contains message buffer configuration data for the transmission mode, the channel assignment, and for the cycle counter filtering. For detailed information on cycle counter filtering, refer to Message Buffer Cycle Counter Filtering .

NOTE

If at least one message buffer assigned to a certain slot is assigned to both channels, then all message buffers assigned to this slot have to be assigned to both channels. Otherwise, the message buffer configuration is illegal and the result of the message buffer search is not defined.

Table 50-17. Channel Assignment Description

CHA	CHB	Transmit Message Buffer		Receive Message Buffer	
		static segment	dynamic segment	static segment	dynamic segment
1	1	transmit on both channel A and channel B	transmit on channel A only	store first valid frame received on either channel A or channel B	store first valid frame received on channel A, ignore channel B
0	1	transmit on channel B	transmit on channel B	store first valid frame received on channel B	store first valid frame received on channel B
1	0	transmit on channel A	transmit on channel A	store first valid frame received on channel A	store first valid frame received on channel A
0	0	no frame transmission	no frame transmission	no frame stored	no frame stored

Address: 0h base + 802h offset + (8d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read Write	MTM	CHA	CHB	CCFE	CCFMSK			
Reset	0*	0*	0*	0*	0*	0*	0*	0*

Bit	8	9	10	11	12	13	14	15
Read Write	CCFMSK		CCFVAL					
Reset	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBCCFR*n* field descriptions

Field	Description
0 MTM	Message Buffer Transmission Mode. This control bit applies only to transmit message buffers and defines the transmission mode. 0 Event transmission mode 1 State transmission mode
1 CHA	Channel Assignment. In conjunction with the CHB field, defines the channel assignment and control the receive and transmit behavior of the message buffer according to Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*) .
2 CHB	Channel Assignment. In conjunction with the CHB field, defines the channel assignment and control the receive and transmit behavior of the message buffer according to Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*) .

Table continues on the next page...

FR_MBCCFR*n* field descriptions (continued)

Field	Description
3 CCFE	Cycle Counter Filtering Enable. This control bit is used to enable and disable the cycle counter filtering. 0 Cycle counter filtering disabled 1 Cycle counter filtering enabled
4–9 CCFMSK	Cycle Counter Filtering Mask. This field defines the filter mask for the cycle counter filtering.
10–15 CCFVAL	Cycle Counter Filtering Value. This field defines the filter value for the cycle counter filtering.

50.6.114 Message Buffer Frame ID Register (FR_MBFIDR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Address: 0h base + 804h offset + (8d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FID										
Write	▨															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBFIDR*n* field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FID	Frame ID. The semantic of this field depends on the message buffer transfer type. • Receive Message Buffer: This field is used as a filter value to determine if the message buffer is used for reception of a message received in a slot with the slot ID equal to FID. • Transmit Message Buffer: This field is used to determine the slot in which the message in this message buffer should be transmitted.

50.6.115 Message Buffer Index Register (FR_MBIDXR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Address: 0h base + 806h offset + (8d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0									MBIDX						
Write	▨															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBIDXR*n* field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBIDX	Message Buffer Index. This field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer. The application writes the index of the initially associated message buffer header field into this register. The CC updates this register after frame reception or transmission. Legal Values are 0 <= i <= 67 . Illegal values will be detected during the message buffer search.

50.6.116 Message Buffer Data Field Offset Register (FR_MBDOR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Address: 0h base + 1000h offset + (2d × i), where i=0d to 67d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MBDO															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBDOR*n* field descriptions

Field	Description
0–15 MBDO	Message Buffer Data Field Offset. This field provides the data field offset belonging to a particular Message Buffer Index. For configuration constraints see Configure Data Field Offsets .

50.6.117 LRAM ECC Error Test Register (FR_LEETR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Address: 0h base + 1090h offset + (2d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	LEETD															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_LEETR*n* field descriptions

Field	Description
0–15 LEETD	LRAM ECC Error Test Data. This field contains the LRAM data belonging to the test register located in LRAM Bank n.

50.7 Functional Description

This section provides a detailed description of the functionality implemented in the CC.

50.7.1 Message Buffer Concept

The CC uses a data structure called a *message buffer* to store frame data, configuration, control, and status data. Each message buffer consists of two parts, the *message buffer control data* and the *physical message buffer*. The message buffer control data are located in dedicated registers. The structure of the message buffer control data depends on the message buffer type and is described in the Message Buffer Types section. The physical message buffer is located in the FlexRay memory area and is described in the next section.

50.7.2 Physical Message Buffer

All FlexRay messages and related frame and slot status information of received frames and of frames to be transmitted to the FlexRay bus are stored in data structures called *physical message buffers*. The physical message buffers are located in the FlexRay memory area.The structure of a physical message buffer is depicted in the figure below.

A physical message buffer consists of two fields, the *message buffer header field* and the *message buffer data field*. The message buffer header field contains the *frame header* and the *slot status*. The message buffer data field contains the *frame data*.

The connection between the two fields is established by the *data field offset*.

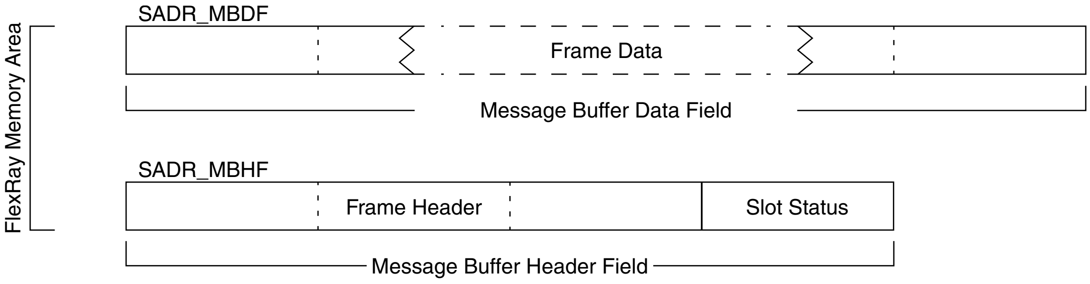


Figure 50-2. Physical Message Buffer Structure

50.7.2.1 Message Buffer Header Field

The message buffer header field is a contiguous region in the FlexRay memory area and occupies eight bytes. It contains the frame header, and the slot status. Its structure is shown in the above figure. The physical start address *SADR_MBHF* of the message buffer header field must be 16-bit aligned.

50.7.2.1.1 Frame Header

The frame header occupies the first six bytes in the message buffer header field. It contains all FlexRay frame header related information according to the *FlexRay Communications System Protocol Specification, Version*2.1 Rev A. A detailed description of the usage and the content of the frame header is provided in Frame Header Description section.

50.7.2.1.2 Slot Status

The slot status occupies the last two bytes of the message buffer header field. It provides the slot and frame status related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the content and usage of the slot status is provided in the Slot Status Description section.

50.7.2.2 Message Buffer Data Field

The message buffer data field is a contiguous area of 2-byte entities. This field contains the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum length of this field depends on the specific message buffer configuration and is specified in the message buffer descriptions given in the following section.

50.7.3 Message Buffer Types

The CC provides three different types of message buffers.

- *Individual Message Buffers*
- *Receive Shadow Buffers*
- *Receive FIFO Buffers*

For each message buffer type the structure of the physical message buffer is identical. The message buffer types differ only in the structure and content of message buffer control data, which control the related physical message buffer. The message buffer control data are described in the following sections.

50.7.3.1 Individual Message Buffers

The individual message buffers are used for all types of frame transmission and for dedicated frame reception based on individual filter settings for each message buffer. The CC supports three types of individual message buffers, which are described in the Individual Message Buffer Functional Description section.

Each individual message buffer consists of two parts, the physical message buffer, which is located in the FlexRay memory area, and the message buffer control data, which are located in dedicated registers. The structure of an individual message buffer is given in the figure below.

Each individual message buffer has a message buffer number n assigned, which determines the set of message buffer control registers associated to this individual message buffer. The individual message buffer with message buffer number *n* is controlled by the registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, and FR_MBIDXRn.

The connection between the message buffer control registers and the physical message buffer is established by the message buffer index field MBIDX in the Message Buffer Index Registers (FR_MBIDXRn). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to this equation:

$$\mathrm{SADR\_MBHF} = (\mathrm{FR\_MBIDXRn}[\mathrm{MBIDX}] \times 8) + \mathrm{SMBA}$$

Equation 17

The data field belonging to a particular physical message buffer is characterized by the data field offset. For each physical message buffer with MBIDX i the FR_MBDORi contains the offset of the corresponding message buffer data field with respect to the CC FlexRay memory area base address as provided by SMBA field in the System Memory Base Address Register (FR_SYMBADR).

The data field offset is used to determine the start address SADR_MBDF of the corresponding message buffer data field in the FlexRay memory area according to this equation:

$$SADR\_MBDF = [DataFieldOffset] + SMBA$$

Equation 18

The FR_MBDORn are stored in the module internal memory LRAM. Refer to CHI LRAM Initialization for the setup of the data field offset values.

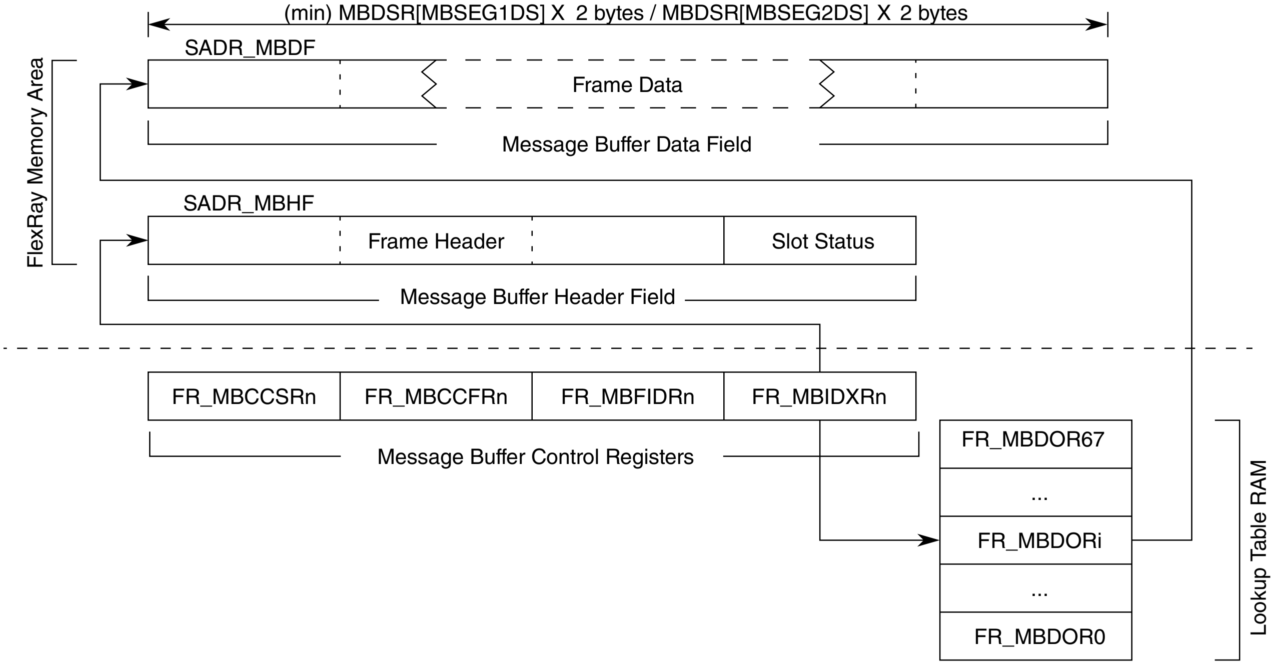


Figure 50-3. Individual Message Buffer Structure

50.7.3.1.1 Individual Message Buffer Segments

The set of the individual message buffers can be split up into two message buffer segments using the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). All individual message buffers with a message buffer number n ≤ FR_MBSSUTR[LAST_MB_SEG1] belong to the first message buffer segment. All individual message buffers with a message buffer number n > FR_MBSSUTR[LAST_MB_SEG1] belong to the second message buffer segment. The following rules apply to the length of the message buffer data field:

- *all physical message buffers associated to individual message buffers that belong to the same message buffer segment must have message buffer data fields of the same length*
- *the minimum length of the message buffer data field for individual message buffers in the first message buffer segment is 2 × FR_MBDSR[MBSEG1DS] bytes*
- *the minimum length of the message buffer data field for individual message buffers assigned to the second segment is 2 × FR_MBDSR[MBSEG2DS] bytes.*

50.7.3.2 Receive Shadow Buffers

The receive shadow buffers are required for the frame reception process for individual message buffers. The CC provides four receive shadow buffers, one receive shadow buffer per channel and per message buffer segment.

Each receive shadow buffer consists of two parts, the physical message buffer located in the FlexRay memory area and the receive shadow buffer control registers located in dedicated registers. The structure of a receive shadow buffer is shown in the figure below. The four internal shadow buffer control registers can be accessed by the Receive Shadow Buffer Index Register (FR_RSBIR).

The connection between the receive shadow buffer control register and the physical message buffer for the selected receive shadow buffer is established by the receive shadow buffer index field RSBIDX in the Receive Shadow Buffer Index Register (FR_RSBIR). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to this equation:

$$\mathrm{SADR\_MBHF} = (\mathrm{FR\_RSBIR}[\mathrm{RSBIDX}] \times 8) + \mathrm{SMBA}$$

Equation 19

The length required for the message buffer data field depends on the message buffer segment that the receive shadow buffer is assigned to. For the receive shadow buffers assigned to the first message buffer segment, the length must be the same as for the individual message buffers assigned to the first message buffer segment. For the receive shadow buffers assigned to the second message buffer segment, the length must be the same as for the individual message buffers assigned to the second message buffer segment. The receive shadow buffer assignment is described in Receive Shadow Buffer Index Register (FR_RSBIR).

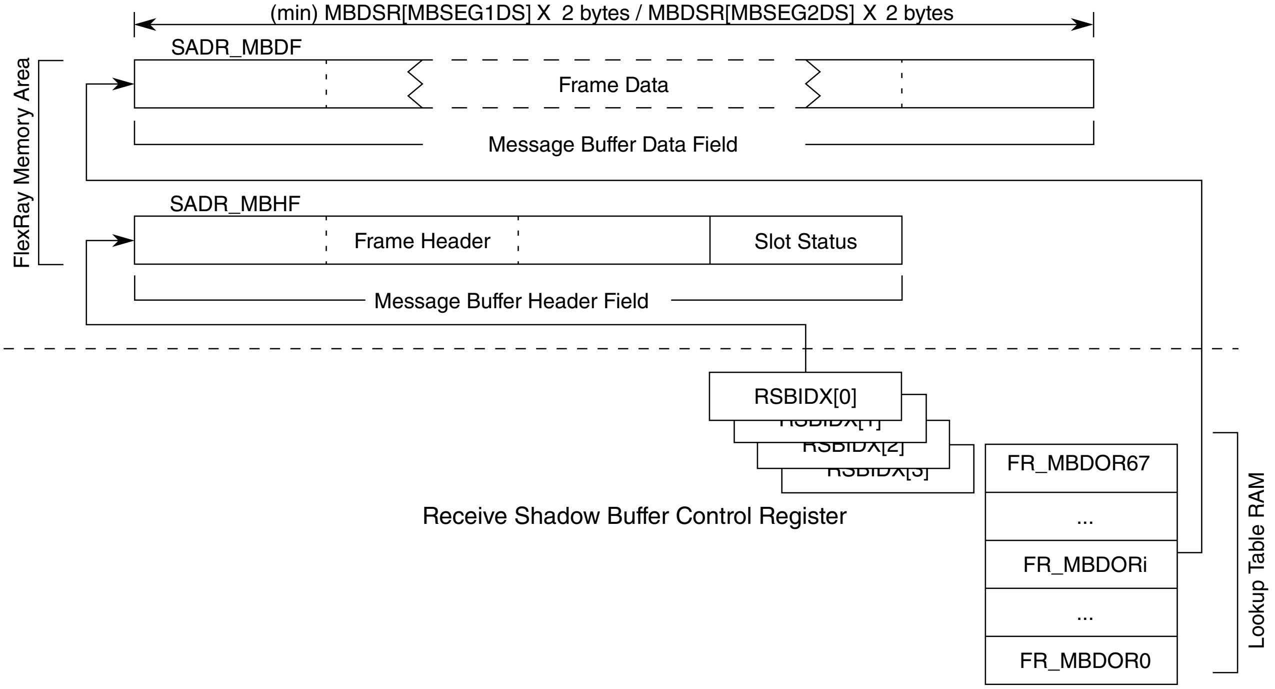


Figure 50-4. Receive Shadow Buffer Structure

50.7.3.3 Receive FIFO

The receive FIFO implements a frame reception system based on the FIFO concept. The CC provides two independent receive FIFOs, one per channel.

A receive FIFO consists of a set of physical message buffers in the FlexRay memory area and a set of receive FIFO control registers located in dedicated registers. The structure of a receive FIFO is given in the Receive FIFO Structure figure, later in this section.

The connection between the receive FIFO control registers and the set of physical message buffers is established by the Receive FIFO Start Index Register (FR_RFSIR), the Receive FIFO Depth and Size Register (FR_RFDSR), and the Receive FIFO A Read Index Register (FR_RFARIR) / Receive FIFO B Read Index Register (FR_RFBRIR).

The system memory base address SMBA valid for the receive FIFOs is defined by the system memory base address register selected by the FIFO address mode bit FR_MCR[FAM] in the Module Configuration Register.

The start byte address SADR_MBHF[1] of the first message buffer header field that belongs to the receive FIFO is determined according to Equation 20 on page 1778.

$$\mathrm{SADR\_MBHF}[1] = (8 \times \mathrm{FR\_RFSIR}[\mathrm{SIDX}]) + \mathrm{SMBA}$$

Equation 20

The start byte address SADR_MBHF[n] of the last message buffer header field that belongs to the receive FIFO in the FlexRay memory area is determined according Equation 21 on page 1779 .

$$\mathrm{SADR\_MBHF}[n] = (8 \times (\mathrm{FR\_RFSIR}[\mathrm{SIDX}] + \mathrm{FR\_RFDSR}[\mathrm{FIFO\_DEPTH}])) + \mathrm{SMBA}$$

Equation 21

The required information to access the current entry of the FIFO is given in the following registers:

- *The registers Receive FIFO A Read Index Register (FR_RFARIR) and Receive FIFO B Read Index Register (FR_RFBRIR) provide the index of the physical message buffer belonging to the current entry.*

  The data field offset belonging to the current FIFO entry RF_DFO[X] must be calculated using the current read index i according to the following formula:

$$\mathrm{RF\_DFO}[X] = \mathrm{FR\_RFSDOR}[X] + (\mathrm{FR\_RFDSR}[X][\mathrm{ENTRY\_SIZE}] \times 2) \times i - \mathrm{FR\_RFSIDX}[X\})$$

Equation 22

Note

The current read index loops up starting at the number given in the FR_RD[A/B]RDIDX register for the required number of entries.

Refer to FIFO Update for details about updating the FIFO read pointer.

All message buffer header fields assigned to a receive FIFO are within a contiguous region defined by FR_RFSIR[SIDX] and FR_RFDSR[FIFO_DEPTH].

The data sections of all FIFO entries within on receive FIFO are of the same length defined by FR_RFDSR[FIFO_SIZE].

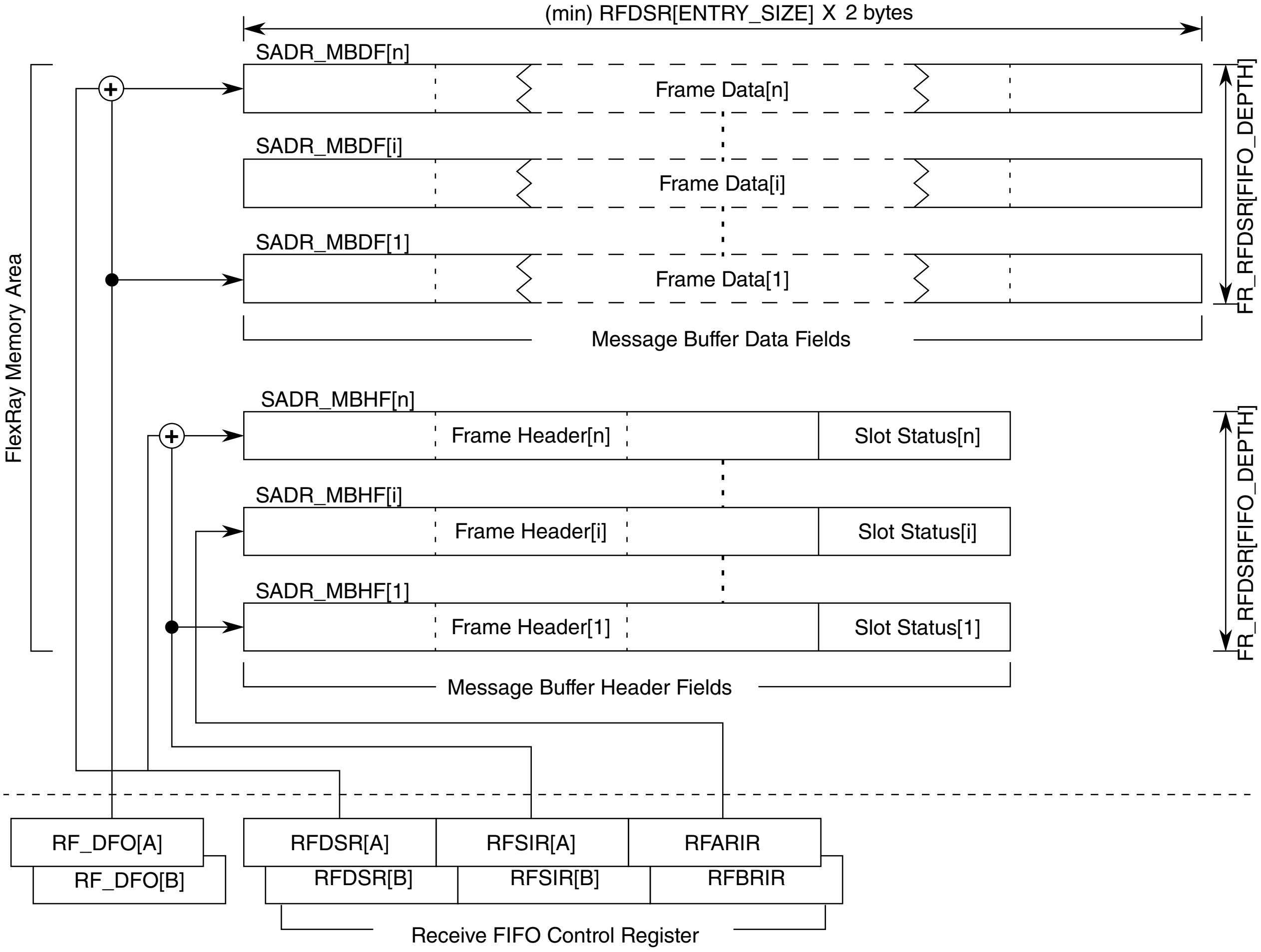


Figure 50-5. Receive FIFO Structure

Note

The actual values of the data field offsets RF_DFO[A/B] need to be calculated according to Equation 22 on page 1779. They are not stored in a register.

50.7.3.4 Message Buffer Configuration and Control Data

This section describes the configuration and control data for each message buffer type.

50.7.3.4.1 Individual Message Buffer Configuration Data

Before an individual message buffer can be used for transmission or reception, it must be configured. There is a set of common configuration parameters that applies to all individual message buffers and a set of configuration parameters that applies to each message buffer individually.

50.7.3.4.1.1 Common Configuration Data

The set of common configuration data for individual message buffers is located in the following registers.

- *Message Buffer Data Size Register (FR_MBDSR)*

  The MBSEG2DS and MBSEG1DS fields define the minimum length of the message buffer data field with respect to the message buffer segment.

- *Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)*

  The LAST_MB_SEG1 and LAST_MB_UTIL fields define the segmentation of the individual message buffers and the number of individual message buffers that are used. For more details, see Individual Message Buffer Segments.

50.7.3.4.1.2 Specific Configuration Data

The set of message buffer specific configuration data for individual message buffers is located in the following registers.

- *Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)*

  The MTD bit configures the message buffer type.

- *Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)*

  The MTM, CHA, CHB bits configure the transmission mode and the channel assignment. The CCFE, CCFMSK, and CCFVAL bits and fields configure the cycle counter filter.

- *Message Buffer Frame ID Registers (FR_MBFIDRn)*

  For a transmit message buffer, the FID field is used to determine the slot in which the message in this message buffer will be transmitted.

- *Message Buffer Index Registers (FR_MBIDXRn)This MBIDX field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer.*

50.7.3.5 Individual Message Buffer Control Data

During normal operation, each individual message buffer can be controlled by the control and trigger bits CMT, LCKT, EDT, and MBIE in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn).

50.7.3.6 Receive Shadow Buffer Configuration Data

Before frame reception into the individual message buffers can be performed, the receive shadow buffers must be configured. The configuration data are provided by the Receive Shadow Buffer Index Register (FR_RSBIR). For each receive shadow buffer, the application provides the message buffer header index. When the protocol is in the *POC:normal active* or *POC:normal passive* state, the receive shadow buffers are under full CC control.

50.7.3.7 Receive FIFO Control and Configuration Data

This section describes the configuration and control data for the two receive FIFOs.

50.7.3.7.1 Receive FIFO Configuration Data

The CC provides two functional independent receive FIFOs, one per channel. The FIFOs have a common subset of configuration data:

- *Receive FIFO Periodic Timer Register (FR_RFPTR)*

Each FIFO has its own set of configuration data. The configuration data are located in the following registers:

- *Receive FIFO Watermark and Selection Register (FR_RFWMSR)*
- *Receive FIFO Start Index Register (FR_RFSIR)*
- *Receive FIFO Start Data Offset Register (FR_RFSDOR)*
- *Receive FIFO Depth and Size Register (FR_RFDSR)*
- *Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)*
- *Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)*
- *Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)*

- *Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)*
- *Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)*

50.7.3.7.2 Receive FIFO Control Data

The application can access the FIFOs at any time using the control bits in the following registers:

- *Global Interrupt Flag and Enable Register (FR_GIFER)*
- *Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)*

50.7.3.7.3 Receive FIFO Status Data

The current status of the receive fifo is provided in the following register:

- *Global Interrupt Flag and Enable Register (FR_GIFER)*
- *Receive FIFO A Read Index Register (FR_RFARIR)*
- *Receive FIFO B Read Index Register (FR_RFBRIR)*
- *Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)*

50.7.4 FlexRay Memory Area Layout

The CC supports a wide range of possible layouts for the FlexRay memory area. Two basic layout modes can be selected by the FIFO address mode bit FR_MCR[FAM].

50.7.4.1 FlexRay Memory Area Layout (FR_MCR[FAM] = 0)

In the figure given below, shows an example layout for the FIFO address mode FR_MCR[FAM]=0. In this mode, the following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area is one contiguous region.
- The FlexRay memory area size is maximum 64 KB.
- The FlexRay memory area starts at a 16 byte boundary

The FlexRay memory area contains three areas: the *message buffer header area*, the *message buffer data area*, and the *sync frame table area*.

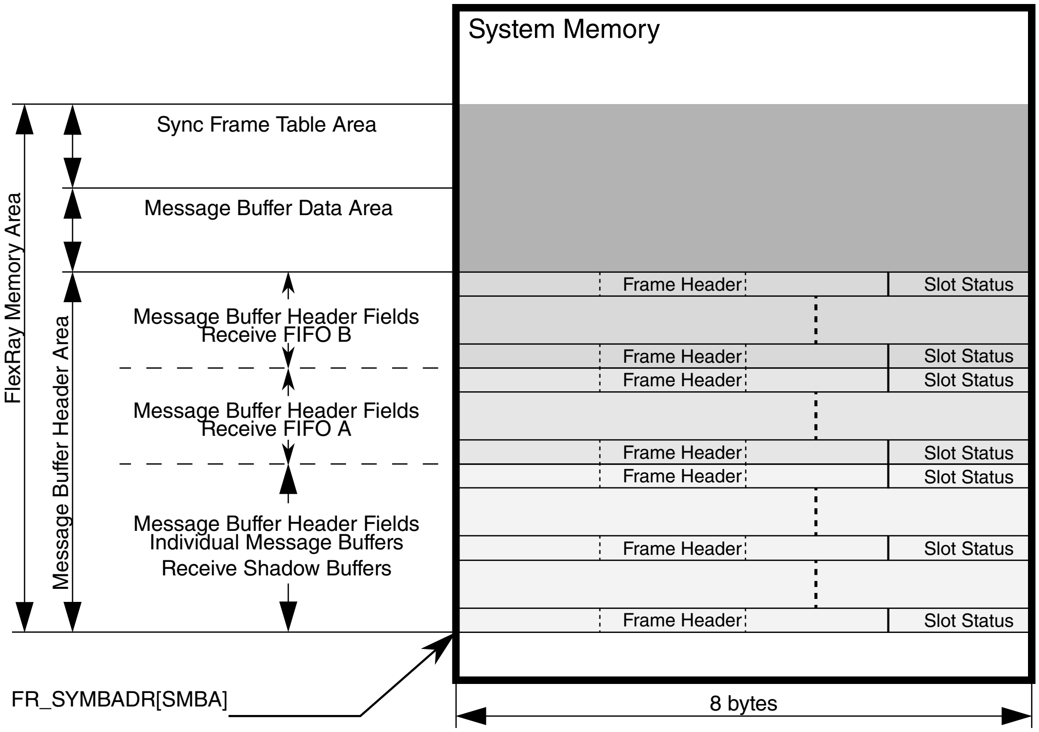


Figure 50-6. Example of FlexRay Memory Area Layout (FR_MCR[FAM] = 0)

50.7.4.2 FlexRay Memory Area Layout (FR_MCR[FAM] = 1)

The following figure shows an example layout for the FIFO address mode FR_MCR[FAM]=1. The following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area consists of two contiguous regions.
- The size of each region is maximum 64 KB.
- Each region start at a 16 byte boundary.

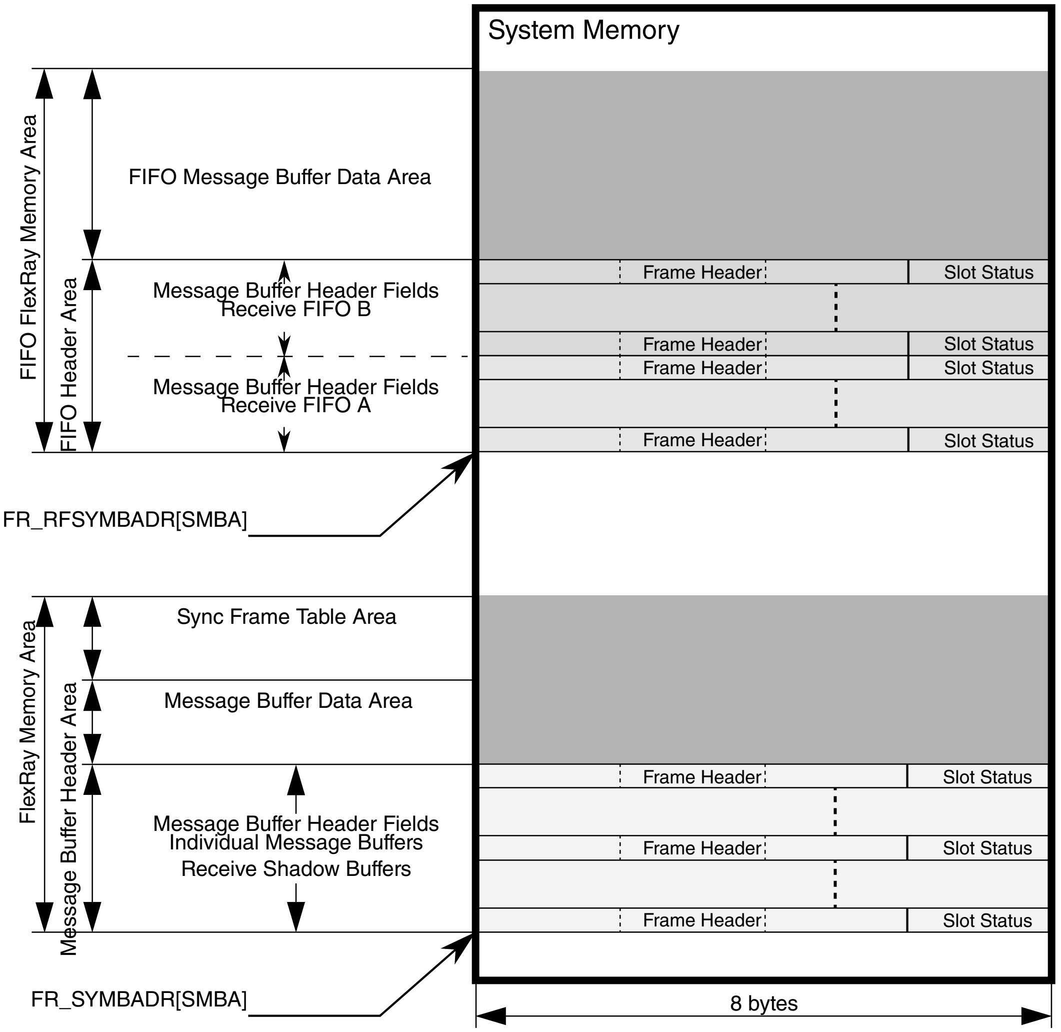


Figure 50-7. Example of FlexRay Memory Area Layout (FR_MCR[FAM] = 1)

50.7.4.3 Message Buffer Header Area (FR_MCR[FAM] = 0)

The message buffer header area contains all message buffer header fields of the physical message buffers for all message buffer types. The following rules apply to the message buffer header fields for the three type of message buffers.

1. The start byte address SADR_MBHF of each message buffer header field for *individual message buffers* and *receive shadow buffers* must fulfill Equation 23 on page 1785.

$$\mathrm{SADR\_MBHF} = (i \times 8) + \mathrm{FR\_SYMBADR[SMBA]}; (0 \le i \le 67)$$

Equation 23

2. The start byte address SADR_MBHF of each message buffer header field for the *FIFO* must fulfill Equation 23 on page 1785.

$$\mathrm{SADR\_MBHF} = (i \times 8) + \mathrm{FR\_SYMBADR[SMBA]}; (0 \leq i \leq 1023)$$

Equation 24

$$\mathrm{SADR\_MBHF} = (i \times 8) + \mathrm{FR\_SYMBADR[SMBA]}; (0 \leq i \leq 1023)$$

Equation 25

3. The message buffer header fields for each FIFO have to be a contiguous area.

50.7.4.4 Message Buffer Header Area (FR_MCR[FAM] = 1)

The message buffer header area contains all message buffer header fields of the physical message buffers for the individual message buffers and receiver shadow buffers. The following rules apply to the message buffer header fields for the two type of message buffers.

1. The start address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill Equation 26 on page 1786.

$$\mathrm{SADR\_MBHF} = (i \times 8) + \mathrm{FR\_SYMBADR[SMBA]}; (0 \leq i \leq 67)$$

Equation 26

50.7.4.5 FIFO Message Buffer Header Area (FR_MCR[FAM] = 1)

The FIFO message buffer header area contains all message buffer header fields of the physical message buffers for the FIFO. The following rules apply to the FIFO message buffer header fields.

1. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill the following equation:Equation 27 on page 1786.

$$\mathrm{SADR\_MBHF} = (i\gamma 8) + \mathrm{FR\_RFSYMBADR[SMBA]}; (0 \leq i \leq 1023)$$

Equation 27

2. The message buffer header fields for each FIFO have to be a contiguous area.

50.7.4.6 Message Buffer Data Area

The message buffer data area contains all the message buffer data fields of the physical message buffers. Each message buffer data field must start at a 16-bit boundary.

50.7.4.7 Sync Frame Table Area

The sync frame table area is used to provide a copy of the internal sync frame tables for application access. Refer to Sync Frame ID and Sync Frame Deviation Tables for the description of the sync frame table area.

50.7.5 Physical Message Buffer Description

This section provides a detailed description of the usage and the content of the two parts of a physical message buffer, the message buffer header field and the message buffer data field.

50.7.5.1 Message Buffer Protection and Data Consistency

The physical message buffers are located in the FlexRay memory area. The CC provides no means to protect the FlexRay memory area from uncontrolled or illegal host or other client write access. To ensure data consistency of the physical message buffers, the application must follow the write access scheme that is given in the description of each of the physical message buffer fields.

50.7.5.2 Message Buffer Header Field Description

This section provides a detailed description of the usage and content of the message buffer header field. A description of the structure of the message buffer header fields is given in Message Buffer Header Field. Each message buffer header field consists of two sections: the frame header section and the slot status section.

50.7.5.2.1 Frame Header Description

Frame header content, access, and checks are discussed in this section.

50.7.5.2.1.1 Frame Header Content

The semantic and content of the frame header section depends on the message buffer type.

For individual receive message buffers and receive FIFOs, the frame header receives the frame header data of the *first valid frame* received on the assigned channels.

For receive shadow buffers, the frame header receives the frame header data of the current frame received regardless of whether the frame is valid or not.

For transmit message buffers, the application writes the frame header of the frame to be transmitted into this location. The frame header will be read out when the frame is transferred to the FlexRay bus.

The structure of the frame header in the message buffer header field for receive message buffers and the receive FIFO is given in the figure below. A detailed description is given in Table 50-22.

Table 50-18. Frame Header Structure (Receive Message Buffer and Receive FIFO)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID										
0x2	0	0	CYCCNT						0	PLDLEN						
0x4	0	0	0	0	0	HDCRC										

The structure of the frame header in the message buffer header field for transmit message buffers is given in the following figure. A detailed description is given in Table 50-23. The checks that will be performed are described in Frame Header Checks.

Table 50-19. Frame Header Structure (Transmit Message Buffer)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID [1]										
0x2	▨	▨	CYCCNT						▨	PLDLEN [2]						
0x4	▨	▨	▨	▨	▨	HDCRC										

▨ = not used

[1] checked

[2] checked if not static

The structure of the frame header in the message buffer header field for transmit message buffers assigned to key slot is given in the figure below.

Table 50-20. Frame Header Structure (Transmit Message Buffer for Key Slot)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	R	PPI	NUF	SYF	SUF	FID										
0x2	▨	▨	CYCCNT						▨	PLDLEN						
0x4	▨	▨	▨	▨	▨	HDCRC										

▨ = not used

50.7.5.2.1.2 Frame Header Access

The frame header is located in the FlexRay memory area. To ensure data consistency, the application must follow the write access scheme described below.

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the frame header field.

For transmit message buffers, the application must follow the write access restrictions given in the following table. This table shows the condition under which the application can write to the frame header entries without corrupting the FlexRay message transmission.

Table 50-21. Frame Header Write Access Constraints (Transmit Message Buffer)

Field	Static Segment	Dynamic Segment
FID	*POC:config* or MB_DIS	
PPI, PLDLEN, HDCRC	*POC:config* or MB_DIS or	
		MB_LCK

50.7.5.2.1.3 Frame Header Checks

As shown in Table 50-19 and Table 50-20 not all fields in the message buffer frame header are used for transmission. Some fields in the message buffer frame header are ignored, some are used for transmission, and some of them are checked for correct values. All checks that will be performed are described below.

For message buffers assigned to the key slot, no checks will be performed.

The value of the FID field must be equal to the value of the corresponding Message Buffer Frame ID Registers (FR_MBFIDRn). If the CC detects a mismatch while transmitting the frame header, it will set the frame ID error flag FID_EF in the CHI Error Flag Register (FR_CHIERFR). The value of the FID field will be ignored and replaced by the value provided in the Message Buffer Frame ID Registers (FR_MBFIDRn).

For transmit message buffers assigned to the *static* segment, the PLDLEN value must be equal to the value of the payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19). If this is not fulfilled, the static payload length error flag SPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct frame is generated with payload_length_static payload words and the payload length field in the transmitted frame header set to payload_length_static.

For transmit message buffers assigned to the *dynamic* segment, the PLDLEN value must be less than or equal to the value of the max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24). If this is not fulfilled, the dynamic payload length error flag DPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct dynamic frame is generated with PLDLEN payload words and the payload length field in the frame header set to PLDLEN.

Table 50-22. Frame Header Field Descriptions (Receive Message Buffer and Receive FFO)

Field	Description
R	**Reserved Bit** — This is the value of the *Reserved bit* of the received frame stored in the message buffer
PPI	**Payload Preamble Indicator** — This is the value of the *Payload Preamble Indicator* of the received frame stored in the message buffer.
NUF	**Null Frame Indicator** — This is the value of the *Null Frame Indicator* of the received frame stored in the message buffer.
SYF	**Sync Frame Indicator** — This is the value of the *Sync Frame Indicator* of the received frame stored in the message buffer.
SUF	**Startup Frame Indicator** — This is the value of the *Startup Frame Indicator* of the received frame stored in the message buffer.
FID	**Frame ID** — This is the value of the *Frame ID* field of the received frame stored in the message buffer.
CYCCNT	**Cycle Count** — This is the number of the communication cycle in which the frame stored in the message buffer was received.
PLDLEN	**Payload Length** — This is the value of the *Payload Length* field of the received frame stored in the message buffer.
HDCRC	**Header CRC** — This is the value of the *Header CRC* field of the received frame stored in the message buffer.

Table 50-23. Frame Header Field Descriptions (Transmit Message Buffer)

Field	Description
R	**Reserved Bit** — This bit is not used, the value of the *Reserved bit* is generated internally according to *FlexRay Communications System Protocol Specification, Version*2.1 Rev A.
PPI	**Payload Preamble Indicator** — This bit provides the value of the *Payload Preamble Indicator* for the frame transmitted from the message buffer.
NUF	**Null Frame Indicator** — This bit is not used, the value of the *Null Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
SYF	**Sync Frame Indicator** — This bit is not used, the value of the *Sync Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
SUF	**Startup Frame Indicator** — This bit is not used, the value of the *Startup Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
FID	**Frame ID** — This field is checked as described in Frame Header Checks.
CYCCNT	**Cycle Count** — This field is not used, the value of the transmitted *Cycle Count* field is taken from the internal communication cycle counter.
PLDLEN	**Payload Length** — This field is checked and used as described in Frame Header Checks.

Table continues on the next page...

Table 50-23. Frame Header Field Descriptions (Transmit Message Buffer) (continued)

Field	Description
HDCRC	**Header CRC** — This field provides the value of the *Header CRC* field for the frame transmitted from the message buffer.

50.7.5.2.2 Slot Status Description

The slot status is a read-only structure for the application and a write-only structure for the CC. The meaning and content of the slot status in the message buffer header field depends on the message buffer type.

50.7.5.2.2.1 Receive Message Buffer and Receive FIFO Slot Status Description

This section describes the slot status structure for the individual receive message buffers and receive FIFOs. The content of the slot status structure for receive message buffers depends on the message buffer type and on the channel assignment for individual receive message buffers as given in the next table.

Table 50-24. Receive Message Buffer Slot Status Content

Receive Message Buffer Type	Slot Status Content
Individual Receive Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Table 50-25
Individual Receive Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Table 50-26
Individual Receive Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Table 50-27
Receive FIFO Channel A Message Buffer	see Table 50-26
Receive FIFO Channel B Message Buffer	see Table 50-27

The meaning of the bits in the slot status structure is explained in the Receive Message Buffer Slot Status Field Description table below.

Table 50-25. Receive Message Buffer Slot Status Structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	CH	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 50-26. Receive Message Buffer Slot Status Structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table continues on the next page...

Table 50-26. Receive Message Buffer Slot Status Structure (ChA) (continued)

R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 50-27. Receive Message Buffer Slot Status Structure (ChB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	1	0	0	0	0	0	0	0	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 50-28. Receive Message Buffer Slot Status Field Description

Field	Description
Common Message Buffer Status Bits	
VFB	**Valid Frame on Channel B** — protocol related variable: *vSS!ValidFrame* channel B 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	**Sync Frame Indicator Channel B** — protocol related variable: *vRF!Header!SyFIndicator* channel B 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	**Null Frame Indicator Channel B** — protocol related variable: *vRF!Header!NFIndicator* channel B 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	**Startup Frame Indicator Channel B** — protocol related variable: *vRF!Header!SuFIndicator* channel B 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	**Syntax Error on Channel B** — protocol related variable: *vSS!SyntaxError* channel B 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	**Content Error on Channel B** — protocol related variable: *vSS!ContentError* channel B 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	**Boundary Violation on Channel B** — protocol related variable: *vSS!BViolation* channel B 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
CH	**Channel first valid received** — This status bit applies only to receive message buffers assigned to the static segment and to both channels. It indicates the channel that has received the *first valid* frame in the slot. This flag is set to 0 if no valid frame was received at all in the subscribed slot. 0 first valid frame received on channel A, or no valid frame received at all

Table continues on the next page...

Table 50-28. Receive Message Buffer Slot Status Field Description (continued)

Field	Description
	1 first valid frame received on channel B
VFA	**Valid Frame on Channel A** — protocol related variable: *vSS!ValidFrame* channel A 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	**Sync Frame Indicator Channel A** — protocol related variable: *vRF!Header!SyFIndicator* channel A 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	**Null Frame Indicator Channel A** — protocol related variable: *vRF!Header!NFIndicator* channel A 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	**Startup Frame Indicator Channel A** — protocol related variable: *vRF!Header!SuFIndicator* channel A 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	**Syntax Error on Channel A** — protocol related variable: *vSS!SyntaxError* channel A 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	**Content Error on Channel A** — protocol related variable: *vSS!ContentError* channel A 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	**Boundary Violation on Channel A** — protocol related variable: *vSS!BViolation* channel A 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1

50.7.5.2.2.2 Transmit Message Buffer Slot Status Description

This section describes the slot status structure for transmit message buffers. Only the TCA and TCB status bits are directly related to the transmission process. All other status bits in this structure are related to a receive process that may have occurred. The content of the slot status structure for transmit message buffers depends on the channel assignment as given in the table below.

Table 50-29. Transmit Message Buffer Slot Status Content

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Table 50-30

Table continues on the next page...

Table 50-29. Transmit Message Buffer Slot Status Content (continued)

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Table 50-31
Individual Transmit Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Table 50-32

The meaning of the bits in the slot status structure is described in the Transmit Message Buffer Slot Status Structure Field Descriptions table.

Table 50-30. Transmit Message Buffer Slot Status Structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 50-31. Transmit Message Buffer Slot Status Structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 50-32. Transmit Message Buffer Slot Status Structure (ChB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	0	0	0	0	0	0	0	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 50-33. Transmit Message Buffer Slot Status Structure Field Descriptions

Field	Description
VFB	**Valid Frame on Channel B** — protocol related variable: *vSS!ValidFrame* channel B 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	**Sync Frame Indicator Channel B** — protocol related variable: *vRF!Header!SyFIndicator* channel B 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	**Null Frame Indicator Channel B** — protocol related variable: *vRF!Header!NFIndicator* channel B 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1

Table continues on the next page...

Table 50-33. Transmit Message Buffer Slot Status Structure Field Descriptions (continued)

Field	Description
SUB	**Startup Frame Indicator Channel B** — protocol related variable: *vRF!Header!SuFIndicator* channel B 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	**Syntax Error on Channel B** — protocol related variable: *vSS!SyntaxError* channel B 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	**Content Error on Channel B** — protocol related variable: *vSS!ContentError* channel B 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	**Boundary Violation on Channel B** — protocol related variable: *vSS!BViolation* channel B 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCB	**Transmission Conflict on Channel B** — protocol related variable: *vSS!TxConflict* channel B 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1
VFA	**Valid Frame on Channel A** — protocol related variable: *vSS!ValidFrame* channel A 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	**Sync Frame Indicator Channel A** — protocol related variable: *vRF!Header!SyFIndicator* channel A 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	**Null Frame Indicator Channel A** — protocol related variable: *vRF!Header!NFIndicator* channel A 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	**Startup Frame Indicator Channel A** — protocol related variable: *vRF!Header!SuFIndicator* channel A 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	**Syntax Error on Channel A** — protocol related variable: *vSS!SyntaxError* channel A 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	**Content Error on Channel A** — protocol related variable: *vSS!ContentError* channel A 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	**Boundary Violation on Channel A** — protocol related variable: *vSS!BViolation* channel A 0 *vSS!BViolation* = 0

Table continues on the next page...

Table 50-33. Transmit Message Buffer Slot Status Structure Field Descriptions (continued)

Field	Description
	1 *vSS!BViolation* = 1
TCA	**Transmission Conflict on Channel A** — protocol related variable: *vSS!TxConflict* channel A 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1

50.7.5.3 Message Buffer Data Field Description

The message buffer data field is used to store the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum required length of this field depends on the message buffer type that the physical message buffer is assigned to and is given in the table below. The structure of the message buffer data field is given in the next figure.

Table 50-34. Message Buffer Data Field Minimum Length

physical message buffer assigned to	minimum length defined by
Individual Message Buffer in Segment 1	FR_MBDSR[MBSEG1DS]
Receive Shadow Buffer in Segment 1	FR_MBDSR[MBSEG1DS]
Individual Message Buffer in Segment 2	FR_MBDSR[MBSEG2DS]
Receive Shadow Buffer in Segment 2	FR_MBDSR[MBSEG2DS]
Receive FIFO for channel A	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 0)
Receive FIFO for channel B	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 1)

Note

The CC will not access any locations outside the message buffer data field boundaries given in the table above.

Table 50-35. Message Buffer Data Field Structure

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	DATA0 / MID0 / NMV0								DATA1 / MID1 / NMV1							
0x2	DATA2 / NMV2								DATA3 / NMV3							
…	…								…							
0xN-2	DATA N-2								DATA N-1							

The message buffer data field is located in the FlexRay memory area; thus, the CC has no means to control application write access to the field. To ensure data consistency, the application must follow a write and read access scheme.

50.7.5.3.1 Message Buffer Data Field Read Access

For transmit message buffers, the CC will not modify the content of the Message Buffer Data Field. Thus the application can read back the data at any time without any impact on data consistency.

For receive message buffers the application must lock the related receive message buffer and retrieve the message buffer header index from the Message Buffer Index Registers (FR_MBIDXRn). While the message buffer is locked, the CC will not update the Message Buffer Data Field.

For receive FIFOs, the application can read the message buffer indicated by the Receive FIFO A Read Index Register (FR_RFARIR) or the Receive FIFO B Read Index Register (FR_RFBRIR) when the related fill levels in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) indicate an non-empty FIFO.

50.7.5.3.2 Message Buffer Data Field Write Access

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the message buffer data field.

For transmit message buffers, the application must follow the write access restrictions given in the Frame Data Write Access Constraints table below.

Table 50-36. Frame Data Write Access Constraints

Field	CC/MB State
DATA, MID, NMV	*POC:config* or MB_DIS or MB_LCK

Table 50-37. Frame Data Field Descriptions

Field	Description
DATA 0, DATA 1, … DATA N-1	**Message Data** — Provides the message data received or to be transmitted.For receive message buffer and receive FIFOs, this field provides the message data received for this message buffer. For transmit message buffers, the field provides the message data to be transmitted.
MID 0, MID 1	**Message Identifier** — If the payload preamble bit PPI is set in the message buffer frame header, the MID field holds the message ID of a dynamic frame located in the message buffer. The receive FIFO filter uses the received message ID for message ID filtering.

Table continues on the next page...

Table 50-37. Frame Data Field Descriptions (continued)

Field	Description
NMV 0, NMV 1, … NMV 11	**Network Management Vector** — If the payload preamble bit PPI is set in the message buffer frame header, the network management vector field holds the network management vector of a static frame located in the message buffer. **Note:** The MID and NMV bytes replace the corresponding DATA bytes.

50.7.6 Individual Message Buffer Functional Description

The CC provides these basic types of individual message buffers:

1. Transmit Message Buffers
2. Receive Message Buffers

Before an individual message buffer can be used, it must be configured by the application. After the initial configuration, the message buffer can be reconfigured later. The set of the configuration data for individual message buffers is given in Individual Message Buffer Configuration Data.

50.7.6.1 Individual Message Buffer Configuration

The individual message buffer configuration consists of two steps. The first step is the allocation of the required amount of memory for the FlexRay memory area. The second step is the programming of the message buffer configuration registers, which is described in this section.

50.7.6.1.1 Common Configuration Data

One part of the message buffer configuration data is common to all individual message buffers and the receive shadow buffers. These data can only be set when the protocol is in the *POC:config* state.

The application configures the number of utilized individual message buffers by writing the message buffer number of the last utilized message buffer into the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the size of the two segments of individual message buffers by writing the message buffer number of the last message buffer in the first segment into the LAST_MB_SEG1 field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the length of the message buffer data fields for both of the message buffer segments by writing to the MBSEG2DS and MBSEG1DS fields in the Message Buffer Data Size Register (FR_MBDSR).

Depending on the current receive functionality of the CC, the application must configure the receive shadow buffers. For each segment and for each channel with at least one individual receive message buffer assigned, the application must configure the related receive shadow buffer using the Receive Shadow Buffer Index Register (FR_RSBIR).

50.7.6.1.2 Specific Configuration Data

The second part of the message buffer configuration data is specific for each message buffer.

These data can be changed only when either

- *the protocol is in the POC:config state or*
- *the message buffer is disabled, i.e. FR_MBCCSRn[EDS] = 0*

The individual message buffer type is defined by the MTD and MBT bits in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) as given in the following table.

Table 50-38. Individual Message Buffer Types

FR_MBCCSRn		Individual Message Buffer Description
MTD	MBT	
0	0	Receive Message Buffer
0	1	Reserved
1	0	Transmit Message Buffer
1	1	Reserved

The message buffer specific configuration data are

1. MTD bits in Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)
2. all fields and bits in Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)

3. all fields and bits in Message Buffer Frame ID Registers (FR_MBFIDRn)
4. all fields and bits in Message Buffer Index Registers (FR_MBIDXRn)

The meaning of the specific configuration data depends on the message buffer type, as given in the detailed message buffer type descriptions in the Transmit Message Buffers section and Receive Message Buffers section.

50.7.6.2 Transmit Message Buffers

The section provides a detailed description of the functionality of single buffered transmit message buffers.

A transmit message buffer is used by the application to provide message data to the CC that will be transmitted over the FlexRay Bus. The CC uses the transmit message buffers to provide information about the transmission process and status information about the slot in which message was transmitted.

The individual message buffer with message buffer number n is configured to be a transmit message buffer by the following settings:

- FR__ MBCCSRn[MBT] = 0(singlebufferedmessagebuffer)
- FR_ MBCCSRn[MDT] = 1(transmitmessagebuffer)

50.7.6.2.1 Access Regions

To certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is implemented, which is used to control the access to the data, control, and status bits of a message buffer. The access regions for transmit message buffers are depicted in the figure below. A description of the regions is given in Table 50-40. If an region is active as indicated in Table 50-41, the access scheme given for that region applies to the message buffer.

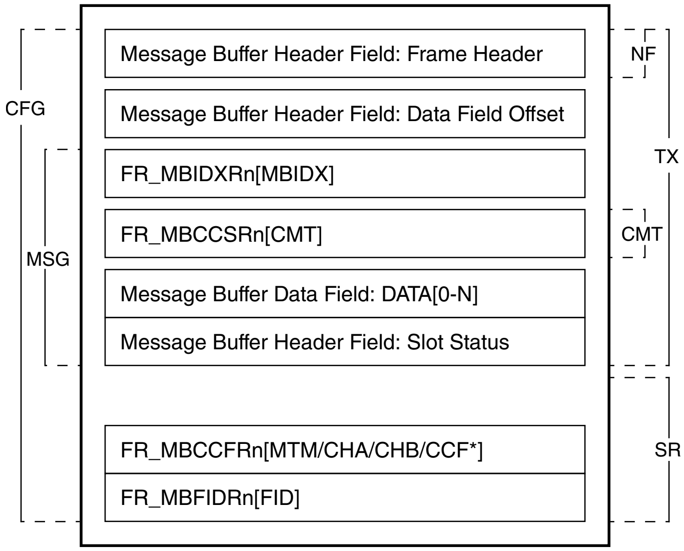

CFG
MSG
Message Buffer Header Field: Frame Header
Message Buffer Header Field: Data Field Offset
FR_MBIDXRn[MBIDX]
FR_MBCCSRn[CMT]
Message Buffer Data Field: DATA[0-N]
Message Buffer Header Field: Slot Status
FR_MBCCFRn[MTM/CHA/CHB/CCF*]
FR_MBFIDRn[FID]
NF
TX
CMT
SR

Figure 50-8. Transmit Message Buffer Access Regions

Table 50-39. Transmit Message Buffer Access Regions Description

Region	Access from		Region used for
	Application	Module	
CFG	read/write	-	Message Buffer Configuration
MSG	read/write	-	Message Data and Slot Status Access
NF	-	read-only	Message Header Access for Null Frame Transmission
TX	-	read/write	Message Transmission and Slot Status Update
CM	-	read-only	Message Buffer Validation
SR	-	read-only	Message Buffer Search

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT], and the interrupt enable bit FR_MBCCSRn[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC clear access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The transmit message buffer states are given in the next figure. A description of the states is given in Table 50-40, which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required message buffer status information. The internal status information is not visible to the application.

50.7.6.2.2 Message Buffer States

This section describes the transmit message buffer states and provides a state diagram.

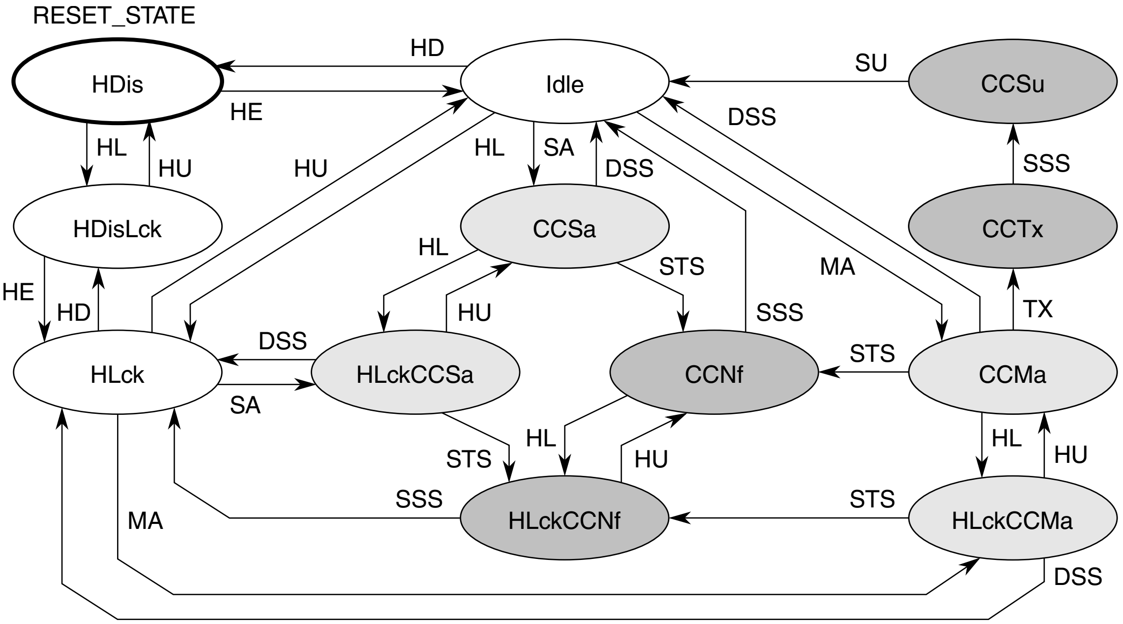


Figure 50-9. Transmit Message Buffer States

Table 50-40. Transmit Message Buffer State Description (Sheet 1 of 2)

State	FR_MBCCSRn		Access Region		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	–	CM,SR	**Idle** - Message Buffer is idle. Included in message buffer search.
HDis	0	0	CFG	–	**Disabled** - Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	–	**Disabled and Locked** - Message Buffer under configuration.Excluded from message buffer search.
HLck	1	1	MSG	SR	**Locked** - Applications access to data, control, and status.Included in message buffer search.
CCSa	1	0	–	–	**Slot Assigned** - Message buffer assigned to next static slot.Ready for Null Frame transmission.
HLckCCSa	1	1	MSG	–	**Locked and Slot Assigned** - Applications access to data, control, and status.Message buffer assigned to next static slot
CCNf	1	0	–	NF	**Null Frame Transmission** Header is used for null frame transmission.

Table continues on the next page...

Table 50-40. Transmit Message Buffer State Description (Sheet 1 of 2) (continued)

State	FR_MBCCSRn		Access Region		Description
	EDS	LCKS	Appl.	Module	
HLckCCNf	1	1	MSG	NF	**Locked and Null Frame Transmission** - Applications access to data, control, and status. Header is used for null frame transmission.
CCMa	1	0	–	CM	**Message Available** - Message buffer is assigned to next slot and cycle counter filter matches.
HLckCCMa	1	1	MSG	–	**Locked and Message Available** - Applications access to data, control, and status. Message buffer is assigned to next slot and cycle counter filter matches.
CCTx	1	0	–	TX	**Message Transmission** - Message buffer data transmit. Payload data from buffer transmitted
CCSu	1	0	–	TX	**Status Update** - Message buffer status update. Update of status flags, the slot status field, and the header index.

50.7.6.2.3 Message Buffer Transitions

Application transitions, module transitions, and transition priorities are discussed in this section.

50.7.6.2.3.1 Application Transitions

The application transitions can be triggered by the application using the commands described in the table below. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

50.7.6.2.3.1.1 Message Buffer Enable and Disable

The enable and disable commands issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in one of the states CCSa, HLckCCSa, CCMa, HLckCCMa, CCNf, HLckCCNf, or CCTx, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

50.7.6.2.3.1.2 Message Buffer Lock and Unlock

The lock and unlock commands issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL and the message buffer is in the state CCTx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 50-41. Transmit Message Buffer Application Transitions

Transition	Command	Condition	Description
HE	FR_MBCCSRn[EDT]:= 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.
HL	FR_MBCCSRn[LCKT]:= 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

50.7.6.2.3.2 Module Transitions

The module transitions that can be triggered by the CC are described in the following table. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 50-42. Transmit Message Buffer Module Transitions

Transition	Condition	Description
SA	slot match andstatic slot	**Slot Assigned** - Message buffer is assigned to next static slot.
MA	slot match andCycleCounter match	**Message Available** - Message buffer is assigned to next slot and cycle counter filter matches.
TX	slot start and FR_MBCCSRn[CMT] = 1	**Transmission Slot Start** - Slot Start and commit bit CMT is set. In case of a dynamic slot, pLatestTx is not exceeded.
SU	status updated	**Status Updated** - Slot Status field and message buffer status flags updated. Interrupt flag set.
STS	static slot start	**Static Slot Start** - Start of static slot.
DSS	dynamic slot start orsymbol window start orNIT start	**Dynamic Slot or Segment Start.** - Start of dynamic slot or symbol window or NIT.
SSS	slot start orsymbol window start orNIT start	**Slot or Segment Start** - Start of static slot or dynamic slot or symbol window or NIT.

50.7.6.2.3.3 Transition Priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the first part of the next table, the module transitions have a higher priority than the application transitions. For all states except the CCMa state, both a lock/unlock transition HL/HD and a module transition can be executed at the same time. The result state is reached by first applying the application transition and subsequently the module transition to the intermediately reached state. For example, if the message buffer is in the HLck state and the application unlocks the message buffer by the HU transition and the module triggers the slot assigned transition SA, the intermediate state is Idle and the resulting state is CCSa.

The priorities among the module transitions is given in the second part of the table below.

Table 50-43. Transmit Message Buffer Transition Priorities

State	Priorities	Description
module vs. application		
Idle, HLck	SA > HD	Slot Assigned > Message Buffer Disable
	MA > HD	Message Available > Message Buffer Disable
CCMa	TX > HL	Transmission Start > Message Buffer Lock
module internal		
Idle, HLck	MA > SA	Message Available > Slot Assigned
CCMa	TX > STS	Transmission Slot Start > Static Slot Start
	TX > DSS	Transmission Slot Start > Dynamic Slot Start

50.7.6.2.4 Transmit Message Setup

To transmit a message over the FlexRay bus, the application writes the message data into the message buffer data field and sets the commit bit CMT in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). The physical access to the message buffer data field is described in Individual Message Buffers.

As indicated by Table 50-33, the application shall write to the message buffer data field and change the commit bit CMT only if the transmit message buffer is in one of the states HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa. The application can change the state of a message buffer if it issues the appropriate commands shown in Table 50-33. The state change is indicated through the FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] status bits.

If the transmit message buffer enters one of the states HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa the FR_MBCCSRn[DVAL] flag is negated.

50.7.6.2.5 Message Transmission

As a result of the message buffer search described in Individual Message Buffer Search, the CC triggers the message available transition MA for up to two transmit message buffers. This changes the message buffer state from Idle to CCMa and the message buffers can be used for message transmission in the next slot.

The CC transmits a message from a message buffer if both of the following two conditions are fulfilled at the start of the transmission slot:

1. the message buffer is in the message available state CCMa
2. the message data are still valid, i.e. FR_MBCCSRn[CMT] = 1

In this case, the CC triggers the TX transition and changes the message buffer state to CCTx. A transmit message buffer timing and state change diagram for message transmission is given in the "Message Transmission Timing" figure below. In this example, the message buffer with message buffer number n is Idle at the start of the search slot, matches the slot and cycle number of the next slot, and message buffer data are valid, i.e. FR_MBCCSRn[CMT] = 1.

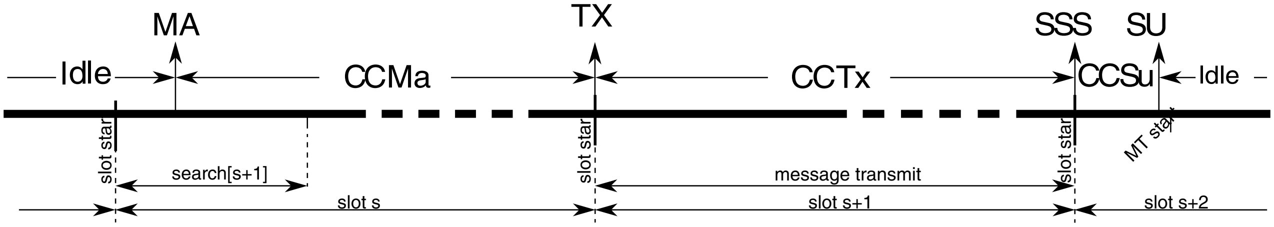


Figure 50-10. Message Transmission Timing

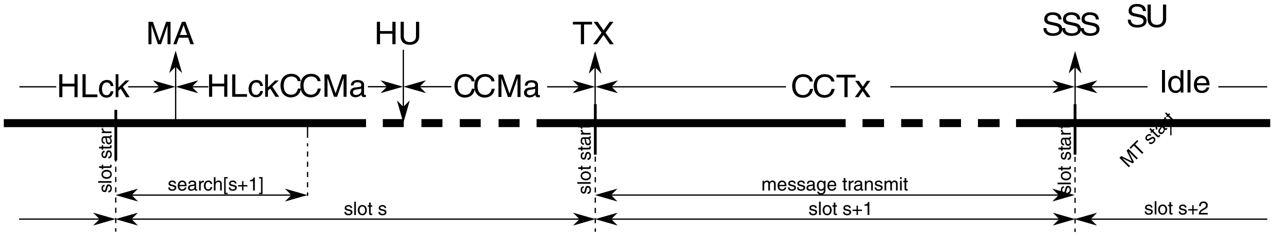


Figure 50-11. Message Transmission from HLck state with unlock

The amount of message data read from the FlexRay memory area and transferred to the FlexRay bus is determined by the following three items

1. the message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. the message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)

3. the value of the PLDLEN field in the message buffer header field, as described in Frame Header Description

If a message buffer is assigned to message buffer segment 1, and PLDLEN > MBSEG1DS, then 2 × MBSEG1DS bytes will be read from the message buffer data field and zero padding is used for the remaining bytes for the FlexRay bus transfer. If PLDLEN <= MBSEG1DS, the CC reads and transfers 2 ×PLDLEN bytes. The same holds for segment 2 and MBSEG2DS.

50.7.6.2.6 Null Frame Transmission

A static slot with slot number S is assigned to the CC for channel A, if at least one transmit message buffer is configured with the FR_MBFIDRn[FID] set to S and FR_MBCCFRn[CHA] set to 1. A Null Frame is transmitted in the static slot S on channel A, if this slot is assigned to the CC for channel A, and all transmit message buffers with FR_MBFIDRn[FID] = s and FR_MBCCFRn[CHA] = 1 are either not committed, i.e FR_MBCCSRn[CMT] = 0, or locked by the application, i.e. FR_MBCCSRn[LCKS] = 1, or the cycle counter filter is enabled and does not match.

Additionally, the application can clear the commit bit of a message buffer that is in the CCMa state, which is called *uncommit* or *transmit abort*. This message buffer will be used for null frame transmission.

As a result of the message buffer search described in Individual Message Buffer Search, the CC triggers the slot assigned transition SA for up to two transmit message buffers if at least one of the conditions mentioned above is fulfilled for these message buffers. The transition SA changes the message buffer states from either Idle to CCSa or from HLck to HLckCCSa. In each case, these message buffers will be used for null frame transmission in the next slot. A message buffer timing and state change diagram for null frame transmission from Idle state is given in the figure below.

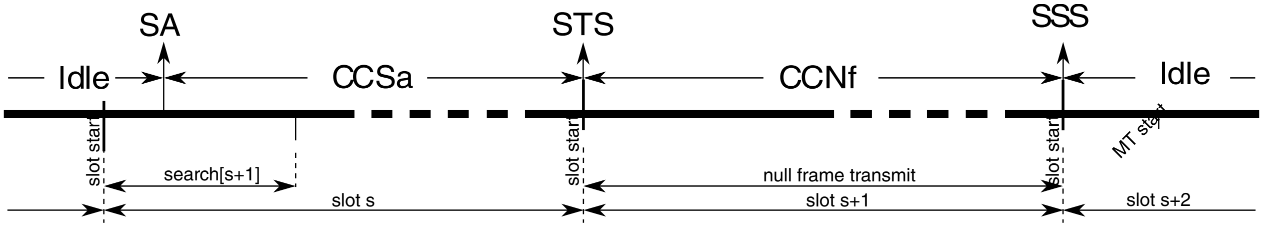


Figure 50-12. Null Frame Transmission from Idle state

A message buffer timing and state change diagram for null frame transmission from HLck state is given in the following figure.

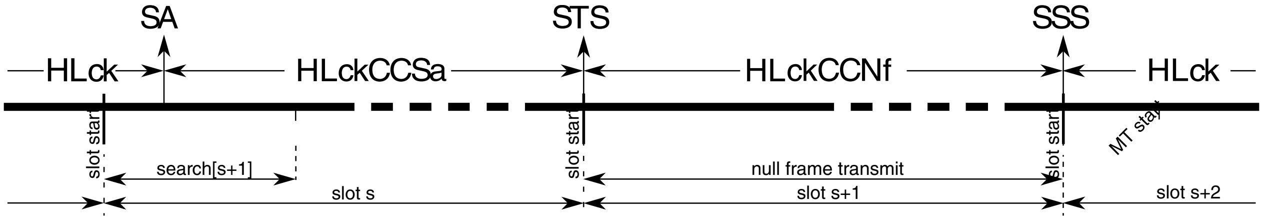


Figure 50-13. Null Frame Transmission from HLck state

If a transmit message buffer is in the CCSa or HLckCCSa state at the start of the transmission slot, a null frame is transmitted in any case, even if the message buffer is unlocked or committed before the transmission slot starts. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in the next figure.

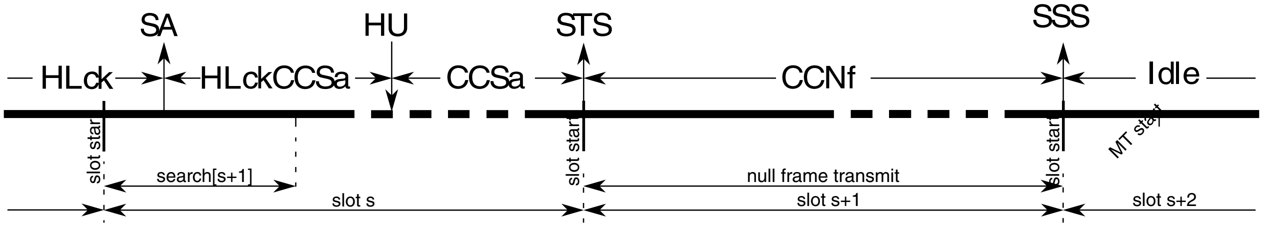


Figure 50-14. Null Frame Transmission from HLck state with unlock

Since the null frame transmission will not use the message buffer data, the application can lock/unlock the message buffer during null frame transmission. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in the figure below.

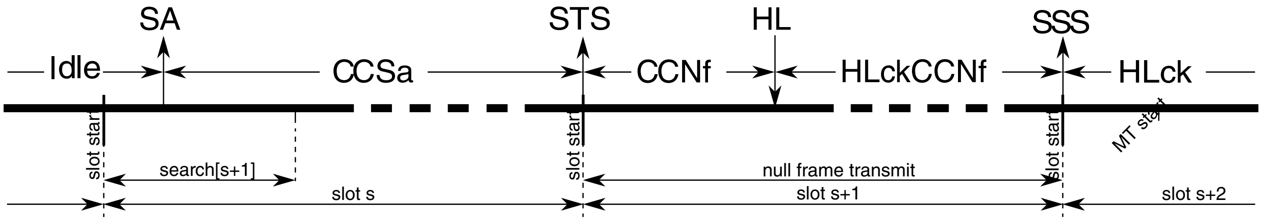


Figure 50-15. Null Frame Transmission from Idle state with locking

50.7.6.2.7 Message Buffer Status Update

After the end of each slot, the PE generates the slot status vector. Depending on the this status, the transmitted frame type, and the amount of transmitted data, the message buffer status is updated.

50.7.6.2.7.1 Message Buffer Status Update after Complete Message Transmission

The term complete message transmission refers to the fact that all payload data stored in the message buffer were send to FlexRay bus. In this case, the CC updates the slot status field of the message buffer and triggers the status updated transition SU. With the SU transition, the CC sets the message buffer interrupt flag FR_MBCCSRn[MBIF] to indicate the successful message transmission.

Depending on the transmission mode flag FR_MBCCFRn[MTM], the CC changes the commit flag FR_MBCCSRn[CMT] and the valid flag FR_MBCCSRn[DVAL]. If the FR_MBCCFRn[MTM] flag is negated, the message buffer is in the *event transmission mode*. In this case, each committed message is transmitted only once. The commit flag FR_MBCCSRn[CMT] is cleared with the SU transition. If the FR_MBCCFRn[MTM] flag is asserted, the message buffer is in the *state transmission mode*. In this case, each committed message is transmitted as long as the application provides new data or locks the message buffers. The CC will not clear the FR_MBCCSRn[CMT] flag at the end of transmission and will set the valid flag FR_MBCCSRn[DVAL] to indicate that the message will be transmitted again.

50.7.6.2.7.2 Message Buffer Status Update after Incomplete Message Transmission

The term incomplete message transmission refers to the fact that not all payload data that should be transmitted were send to FlexRay bus. This may be caused by the following regular conditions in the dynamic segment:

1. The transmission slot starts in a minislot with a minislot number greater than *pLatestTx*.
2. The transmission slot did not exist in the dynamic segment at all.

In any of these two aforementioned conditions occur, the status of the message buffer is not changed at all with the SU transition. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

Additionally, an incomplete message transmission can be caused by internal communication errors. If those errors occur, the Protocol Engine Communication Failure Interrupt Flag PECF_IF is set in the Protocol Interrupt Flag Register 1 (FR_PIFR1).

50.7.6.2.7.3 Message Buffer Status Update after Null Frame Transmission

After the transmission of a null frame, the status of the message buffer that was used for the null frame transmission is not changed at all. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

50.7.6.3 Receive Message Buffers

The section provides a detailed description of the functionality of the receive message buffers. If receive message buffers are used it is required to configure the related receive shadow buffer as described in Receive Shadow Buffers.

A receive message buffer is used to receive a message from the FlexRay Bus based on individual filter criteria. The CC uses the receive message buffer to provide the following data to the application.

1. message data received
2. information about the reception process
3. status information about the slot in which the message was received

A individual message buffer with message buffer number *n* is configured as a receive message buffer by the following configuration settings

$$\mathrm{FR\_\ MBCCSRn}[\mathrm{MTD}] = 0(\mathrm{receivemessagebuffer})$$

Equation 28

To certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is implemented that is used to control the access to the data, control, and status bits of a message buffer. The access regions for receive message buffers are depicted in the figure below. A description of the regions is given in the following table. If a region is active as indicated in the "Receive Message Buffer States and Access (Sheet 2 of 2)" table below, the access scheme given for that region applies to the message buffer.

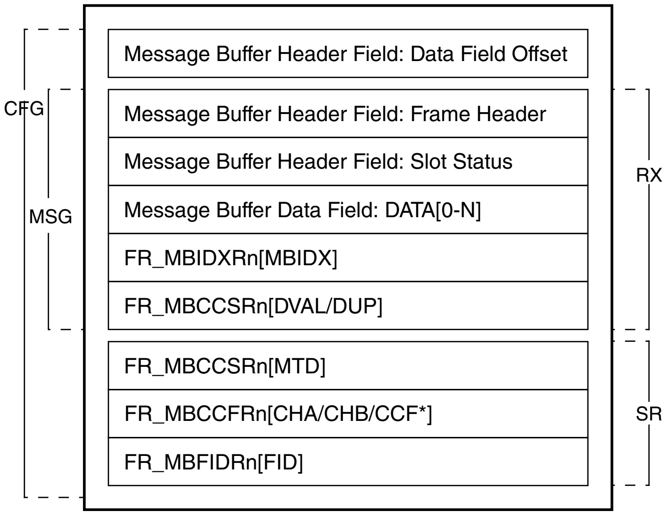

Message Buffer Header Field: Data Field Offset
Message Buffer Header Field: Frame Header
Message Buffer Header Field: Slot Status
Message Buffer Data Field: DATA[0-N]
FR_MBIDXRn[MBIDX]
FR_MBCCSRn[DVAL/DUP]
FR_MBCCSRn[MTD]
FR_MBCCFRn[CHA/CHB/CCF*]
FR_MBFIDRn[FID]
CFG
MSG
RX
SR

Figure 50-16. Receive Message Buffer Access Regions

Table 50-44. Receive Message Buffer Access Region Description

Region	Access from		Region used for
	Application	Module	
CFG	read/write	-	Message Buffer Configuration, Message Data and Status Access
MSG	read/write	-	Message Data, Header, and Status Access
RX	-	write-only	Message Reception and Status Update
SR	-	read-only	Message Buffer Search Data

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT] and the interrupt enable bit FR_MBCCSRn[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC set access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The receive message buffer states are given in the figure below. A description of the message buffer states is given in Table 50-40, which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required status information. The internal status information is not visible to the application.

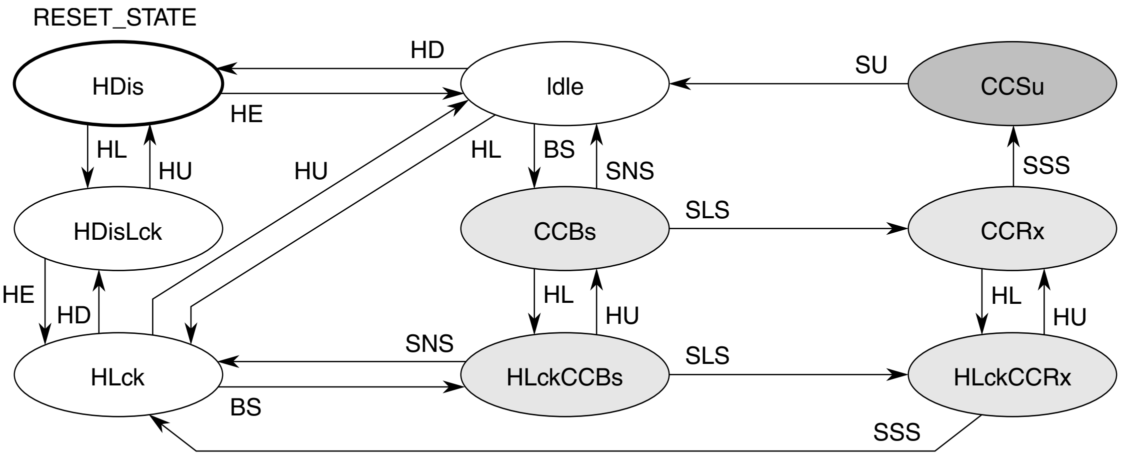


Figure 50-17. Receive Message Buffer States

Table 50-45. Receive Message Buffer States and Access (Sheet 2 of 2)

State	FR_MBCCSRn		Access from		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	–	SR	**Idle** - Message Buffer is idle. Included in message buffer search.
HDis	0	0	CFG	–	**Disabled** - Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	–	**Disabled and Locked** - Message Buffer under configuration.Excluded from message buffer search.
HLck	1	1	MSG	–	**Locked** - Applications access to data, control, and status.Included in message buffer search.
CCBs	1	0	–	–	**Buffer Subscribed** - Message buffer subscribed for reception. Filter matches next (slot, cycle, channel) tuple.
HLckCCBs	1	1	MSG	–	**Locked and Buffer Subscribed** - Applications access to data, control, and status. Message buffer subscribed for reception.
CCRx	1	0	–	–	**Message Receive** - Message data received into related shadow buffer.
HLckCCRx	1	1	MSG	–	**Locked and Message Receive** - Applications access to data, control, and status. Message data received into related shadow buffer.
CCSu	1	0	–	RX	**Status Update** - Message buffer status update. Update of status flags, the slot status field, and the header index.

50.7.6.3.1 Message Buffer Transitions

Application transitions, message buffer enable and disable, message buffer lock and unlock, module transitions, and transition priorities are discussed in this section.

50.7.6.3.1.1 Application Transitions

The application transitions that can be triggered by the application using the commands described in Table 50-41. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

50.7.6.3.1.2 Message Buffer Enable and Disable

The enable and disable commands issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in one of the states CCBs, HLckCCBs, or CCRx, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

50.7.6.3.1.3 Message Buffer Lock and Unlock

The lock and unlock commands issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL while the message buffer is in the state CCRx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 50-46. Receive Message Buffer Application Transitions

Transition	Host Command	Condition	Description
HE	FR_MBCCSRn[EDT]:= 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.
HL	FR_MBCCSRn[LCKT]: = 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

50.7.6.3.1.4 Module Transitions

The module transitions that can be triggered by the CC are described in the next table. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 50-47. Receive Message Buffer Module Transitions

Transition	Condition	Description
BS	slot match and CycleCounter match	**Buffer Subscribed** - The message buffer filter matches next slot and cycle.
SLS	slot start	**Slot Start** - Start of either Static Slot or Dynamic Slot.
SNS	symbol window start or NIT start	**Symbol Window or NIT Start** - Start of either Symbol Window or NIT.
SSS	slot start or symbol window start or NIT start	**Slot or Segment Start** - Start of either Static Slot, Dynamic Slot, Symbol Window, or NIT.
SU	status updated	**Status Updated** - Slot Status field, message buffer status flags, header index updated. Interrupt flag set.

50.7.6.3.1.5 Transition Priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the table below, the module transitions have a higher priority than the application transitions. For all states except the CCRx state, a module transition and the application lock/unlock transition HL/HU and can be executed at the same time. The result state is reached by first applying the module transition and subsequently the application transition to the intermediately reached state. For example, if the message buffer is in the buffer subscribed state CCBs and the module triggers the slot start transition SLS at the same time as the application locks the message buffer by the HL transition, the intermediate state is CCRx and the resulting state is locked buffer subscribed state HLckCCRx.

Table 50-48. Receive Message Buffer Transition Priorities

State	Priorities	Description
module vs. application		
Idle	BS > HD	Buffer Subscribed > Message Buffer Disable
HLck	BS > HD	Buffer Subscribed > Message Buffer Disable
CCRx	SSS > HL	Slot or Segment Start > Message Buffer Lock

50.7.6.3.2 Message Reception

As a result of the message buffer search, the CC changes the state of up to two enabled receive message buffers from either idle state Idle or locked state HLck to the either subscribed state CCBs or locked buffer subscribed state HLckCCBs by triggering the buffer subscribed transition BS.

If the receive message buffers for the next slot are assigned to both channels, then at most one receive message buffer is changed to a buffer subscribed state.

If more than one matching message buffers assigned to a certain channel, then only the message buffer with the lowest message buffer number is in one of the states mentioned above.

With the start of the next static or dynamic slot the module trigger the slot start transition SLS. This changes the state of the subscribed receive message buffers from either CCBs to CCRx or from HLckCCBs to HLckCCRx, respectively.

During the reception slot, the received frame data are written into the shadow buffers. For details on receive shadow buffers, see Receive Shadow Buffers Concept. The data and status of the receive message buffers that are the CCRx or HLckCCRx are not modified in the reception slot.

50.7.6.3.3 Message Buffer Update

With the start of the next static or dynamic slot or with the start of the symbol window or NIT, the module triggers the slot or segment start transition SSS. This transition changes the state of the receiving receive message buffers from either CCRx to CCSu or from HLckCCRx to HLck, respectively.

If a message buffer was in the locked state HLckCCRx, no update will be performed. The received data are lost. This is indicated by setting the Frame Lost Channel A/B Error Flag FRLA_EF/FRLB_EF in the CHI Error Flag Register (FR_CHIERFR).

If a message buffer was in the CCRx state it is now in the CCSu state. After the evaluation of the slot status provided by the PE the message buffer is updated. The message buffer update depends on the slot status bits and the segment the message buffer is assigned to. This is described in Table 50-51.

Table 50-49. Receive Message Buffer Update (Continued)

vSS!ValidFrame	*vRF!Header! NFIndicator*	Update description
1	1	**Valid non-null frame received.** - Message Buffer Data Field updated. - Frame Header Field updated. - Slot Status Field updated. - DUP:= 1 - DVAL:= 1 - MBIF:= 1
1	0	**Valid null frame received.**

Table continues on the next page...

Table 50-49. Receive Message Buffer Update (Continued) (continued)

vSS!ValidFrame	*vRF!Header!NFIndicator*	**Update description**
		- Message Buffer Data Field *not* updated. - Frame Header Field *not* updated. - Slot Status Field updated. - DUP:= 0 - DVAL *not* changed - MBIF:= 1
0	x	**No valid frame received.** - Message Buffer Data Field not updated. - Frame Header Field not updated. - Slot Status Field updated. - DUP:= 0 - DVAL *not* changed. - MBIF:= 1, if the slot was not an empty dynamic slot. **Note:** An empty dynamic slot is indicated by the following frame and slot status bit values: *vSS!ValidFrame* = 0 and *vSS!SyntaxError* = 0 and *vSS!ContentError* = 0 and *vSS!BViolation* = 0.

Note

If the number of the last slot in the current communication cycle on a given channel is n, then all receive message buffers assigned to this channel with FR_MBFIDRn[FID] > *n* will not be updated at all.

When the receive message buffer update has finished the status updated transition SU is triggered, which changes the buffer state from CCSu to Idle. An example receive message buffer timing and state change diagram for a normal frame reception is given in the following figure.

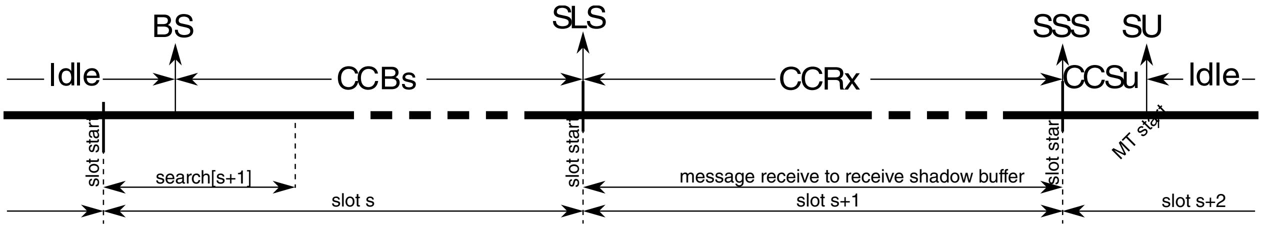


Figure 50-18. Message Reception Timing

The amount of message data written into the message buffer data field of the receive shadow buffer is determined by the following two items:

1. the message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. the message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)
3. the number of bytes received over the FlexRay bus

If the message buffer is assigned to the message buffer segment 1, and the number of received bytes is greater than 2 ×FR_MBDSR.MBSEG1DS, the CC writes only 2 × FR_MBDSR.MBSEG1DS bytes into the message buffer data field of the receive shadow buffer. If the number of received bytes is less than 2 × FR_MBDSR.MBSEG1DS, the CC writes only the received number of bytes and will not change the trailing bytes in the message buffer data field of the receive shadow buffer. The same holds for the message buffer segment 2 with FR_MBDSR.MBSEG2DS.

50.7.6.3.4 Received Message Access

To access the message data received over the FlexRay bus, the application reads the message data stored in the message buffer data field of the corresponding receive message buffer. The access to the message buffer data field is described in Individual Message Buffers.

The application can read the message buffer data field if the receive message buffer is one of the states HDis, HDisLck, or HLck. If the message buffer is in one of these states, the CC will not change the content of the message buffer.

50.7.6.3.5 Receive Shadow Buffers Concept

The receive shadow buffer concept applies only to individual receive message buffers. The intention of this concept is to ensure that only syntactically and semantically valid received non-null frames are presented to the application in a receive message buffer. The basic structure of a receive shadow buffer is described in Receive Shadow Buffers.

The receive shadow buffers temporarily store the received frame header and message data. After the slot boundary the slot status information is generated. If the slot status information indicates the reception of the valid non-null frame (see Table 50-49), the CC writes the slot status into the slot status field of the receive shadow buffer and exchanges the content of the Message Buffer Index Registers (FR_MBIDXRn) with the content of the corresponding internal shadow buffer index register. In all other cases, the CC writes the slot status into the identified receive message buffer, depending on the slot status and the FlexRay segment the message buffer is assigned to.

The shadow buffer concept, with its index exchange, results in the fact that the FlexRay memory area located message buffer associated to an individual receive message buffer changes after successful reception of a valid frame. This means that the message buffer area in the FlexRay memory area accessed by the application for reading the received message is different from the initial setting of the message buffer. Therefore, the application must not rely on the index information written initially into the Message Buffer Index Registers (FR_MBIDXRn). Instead, the index of the message buffer header field must be fetched from the Message Buffer Index Registers (FR_MBIDXRn).

50.7.7 Individual Message Buffer Search

This section provides a detailed description of the message buffer search algorithm.

The message buffer search determines for each enabled channel if a slot s in a communication cycle c is assigned for frame or null frame transmission or if it is subscribed for frame reception on that channel.

The message buffer search is a sequential algorithm which is invoked at the following protocol related events:

1. NIT start
2. slot start in the static segment
3. minislot start in the dynamic segment

The message buffer search within the NIT searches for message buffers assigned or subscribed to slot 1. The message buffer search within slot *n* searches for message buffers assigned or subscribed to slot *n+1*.

In general, the message buffer search for the next slot *n* considers only message buffers which are

1. enabled, i.e. FR_MBCCSRn[EDS] = 1, and
2. matches the next slot *n*, i.e. FR_MBFIDRn[FID] = *n*, and

On top of that, for the static segment only those message buffers are considered, that match the condition of at least one row of the "Message Buffer Search Priority (static segment)" table shown below. For the dynamic segment only those message buffers are considered, that match the condition of at least one row of the next "Message Buffer Search Priority (dynamic segment)" table. These message buffers are called *matching* message buffers.

For each enabled channel the message buffer search may identify multiple *matching* message buffers. Among all matching message buffers the message buffers with highest priority according to the "Message Buffer Search Priority (static segment)" table for the static segment and according to the "Message Buffer Search Priority (dynamic segment)" table for the dynamic segment are selected.

Table 50-50. Message Buffer Search Priority (static segment)

Priority	MTD	LCKS	CMT	CCFM-[1]	Description	Transition
(highest) 0	1	0	1	1	transmit buffer, matches cycle count, not locked and committed	MA
1	1	-	0	1	transmit buffer, matches cycle count, not committed	SA
	1	1	-	1	transmit buffer, matches cycle count, locked	SA
2	1	-	-	-	transmit buffer	SA
3	0	0	n/a	1	receive buffer, matches cycle count, not locked	SB
(lowest) 4	0	1	n/a	1	receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.
2. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.

Table 50-51. Message Buffer Search Priority (dynamic segment)

Priority	MTD	LCKS	CMT	CCFM-[1]	Description	Transition
(highest) 0	1	0	1	1	transmit buffer, matches cycle count, not locked and committed	MA
1	0	0	n/a	1	receive buffer, matches cycle count, not locked	SB
(lowest) 2	0	1	n/a	1	receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.
2. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.

If there are multiple message buffer with highest priority, the message buffer with the lowest message buffer number is selected. All message buffer which have the highest priority must have a consistent channel assignment as specified in Message Buffer Channel Assignment Consistency.

Depending on the message buffer channel assignment the same message buffer can be found for both channel A and channel B. In this case, this message buffer is used as described in Individual Message Buffers.

50.7.7.1 Message Buffer Cycle Counter Filtering

The message buffer cycle counter filter is a value-mask filter defined by the CCFE, CCFMSK, and CCFVAL fields in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn). This filter determines a set of communication cycles in which the message buffer is considered for message reception or message transmission. If the cycle counter filter is disabled, i.e. CCFE = 0, this set of cycles consists of all communication cycles.

If the cycle counter filter of a message buffer does not match a certain communication cycle number, this message buffer is not considered for message transmission or reception in that communication cycle. In case of a transmit message buffer assigned to a slot in the static segment, though, this buffer is added to the matching message buffers to indicate the slot assignment and to trigger the null frame transmission.

The cycle counter filter of a message buffer matches the communication cycle with the number CYCCNT if at least one of the following conditions evaluates to true:

$$\mathrm{MBCCFRn}[\mathrm{CCFE}] = 0$$

$$\mathrm{CYCCNT}\ \&\ \mathrm{MBCCFRn}[\mathrm{CCFMSK}] = \mathrm{MBCCFRn}[\mathrm{CCFVAL}]\ \&\ \mathrm{MBCCFRn}[\mathrm{CCFMSK}]$$

50.7.7.2 Message Buffer Channel Assignment Consistency

The message buffer channel assignment given by the CHA and CHB bits in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn) defines the channels on which the message buffer will receive or transmit. The message buffer with number *n* transmits or receives on channel A if FR_MBCCFRn[CHA] = 1 and transmits or receives on channel B if FR_MBCCFRn[CHB] = 1.

To ensure correct message buffer operation, all message buffers assigned to the same slot and with the same priority must have a *consistent* channel assignment. That means they must be either assigned to one channel only, or must be assigned to *both* channels. The behavior of the message buffer search is not defined, if both types of channel assignments occur for one slot and priority. An inconsistent channel assignment for message buffer 0 and message buffer 1 is depicted in the figure below.

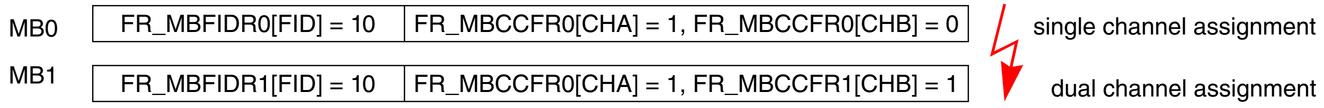

MB0
FR_MBFIDR0[FID] = 10
FR_MBCCFR0[CHA] = 1, FR_MBCCFR0[CHB] = 0
single channel assignment
MB1
FR_MBFIDR1[FID] = 10
FR_MBCCFR0[CHA] = 1, FR_MBCCFR1[CHB] = 1
dual channel assignment

Figure 50-19. Inconsistent Channel Assignment

50.7.7.3 Node Related Slot Multiplexing

The term *Node Related Slot Multiplexing* applies to the dynamic segment only and refers to the functionality if there are transmit as well as receive message buffers are configured for the same slot.

According to Table 50-51 the transmit buffer is only found if the cycle counter filter matches, and the buffer is not locked and committed. In all other cases, the receive buffer will be found. Thus, if the block has no data to transmit in a dynamic slot, it is able to receive frames on that slot.

50.7.7.4 Message Buffer Search Error

There are two kinds of errors which may occur during message buffer search[1].

50.7.7.4.1 Message Buffer Search Start while Running

If the message buffer search is running in slot *n-1* and the next message buffer search start event appears due to the start of slot *n*, the message buffer search engine is stopped and the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR). As a result of this stop, no individual message buffer is identified for transmission or reception in slot *n*. Additionally, the search engine will not be started in slot *n*, and consequently no individual message buffer is identified for transmission or reception in slot *n+1*.

A message buffer search error appears only if the CHI frequency is too slow to allow the search through all message buffers to be completed within the NIT or a minislot.

For more details of minimum required CHI frequency see Number of Usable Message Buffers.

50.7.7.4.2 Illegal Message Buffer Index Found

If the message buffer search has finished the message buffer search in slot *n-1*, it retrieves the data offset values for the found message buffers and the receive shadow buffers. If one of these message buffers contains an illegal message buffer index, the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR) is set and no individual message buffer is identified for transmission or reception in slot

[1] The FIFO reception is not affected by the search errors. Additionally, if no rx buffer has been found due to an search error, the received frame is considered for FIFO reception.

n. The legal message buffer index values for the individual and receive shadow buffers are specified in the Receive Shadow Buffer Index Register section (FR_RSBIR) and the Message Buffer Index Registers sections (FR_MBIDXRn).

50.7.8 Individual Message Buffer Reconfiguration

The initial configuration of each individual message buffer can be changed even when the protocol is not in the *POC:config* state. This is referred to as individual message buffer *reconfiguration*. The configuration bits and fields that can be changed are given in the section on Specific Configuration Data. The common configuration data given in the section on Specific Configuration Data can not be reconfigured when the protocol is out of the *POC:config* state.

50.7.8.1 Reconfiguration Schemes

Depending on the target and destination basic state of the message buffer that is to be reconfigured, there are three reconfiguration schemes.

50.7.8.1.1 Basic Type Not Changed (RC1)

A reconfiguration will not change the basic type of the individual message buffer, if the message buffer transfer direction bit FR_MBCCSRn[MTD] are not changed. This type of reconfiguration is denoted by RC1 in Figure 50-20. Transmit and receive message buffers can be RC1-reconfigured when in the HDis or HDisLck state.

50.7.8.1.2 Buffer Type Not Changed (RC2)

A reconfiguration will not change the buffer type of the individual message buffer. This type of reconfiguration is denoted by RC2 in the figure below. It applies to transmit and receive message buffers. Transmit and receive message buffers can be RC2-reconfigured when in the HDis or HDisLck state.

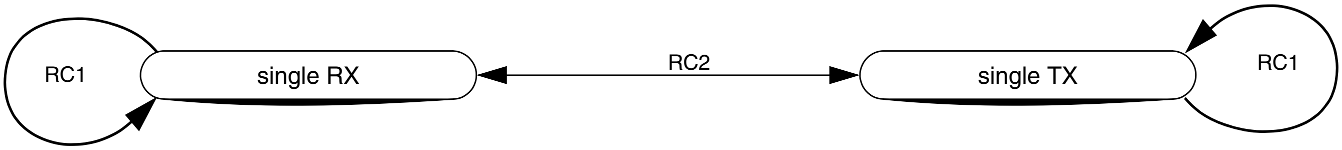


Figure 50-20. Message Buffer Reconfiguration Scheme

50.7.9 Receive FIFOs

This section provides the functional description of the two receive FIFOs.

50.7.9.1 Overview

The two receive FIFOs implement the queued message buffer concept defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. One FIFO is assigned to channel A, the other FIFO is assigned to channel B. Both FIFOs work completely independent from each other.

The message buffer structure of each FIFO is described in Receive FIFO. The area in the FlexRay memory area for each of the two FIFOs is characterized by:

- *The FIFO system memory base address*
- *The index of the first FIFO entry given by Receive FIFO Start Index Register (FR_RFSIR)*
- *The data field offset of the data field belonging to the first FIFO entry given by Receive FIFO Start Data Offset Register (FR_RFSDOR)*
- *The number of FIFO entries and the length of each FIFO entry as given by Receive FIFO Depth and Size Register (FR_RFDSR)*

50.7.9.2 FIFO Configuration

The FIFOs can be configured for two different locations of the system memory base address via the FIFO address mode bit FAM in the Module Configuration Register (FR_MCR).

50.7.9.2.1 Single System Memory Base Address Mode

This mode is configured, when the FIFO address mode flag FR_MCR[FAM] is set to 0. In this mode, the location of the system memory base address for the FIFO buffers is System Memory Base Address Register (FR_SYMBADR).

50.7.9.2.2 Dual System Memory Base Address Mode

This mode is configured, when the FIFO address mode flag FR_MCR[FAM] is set to 1. In this mode, the location of the system memory base address for the FIFO buffers is Receive FIFO System Memory Base Address Register (FR_RFSYMBADR).

The FIFO control and configuration data are given in Receive FIFO Control and Configuration Data. The configuration of the FIFOs consists of two steps.

The first step is the allocation of the required amount of FlexRay memory area for the FlexRay window. This includes the allocation of the message buffer header area and the allocation of the message buffer data fields. For more details see FlexRay Memory Area Layout.

The second step is the programming of the configuration data register while the PE is in *POC:config*.

The following steps configure the layout of the FIFO.

- Configure the FIFO update and address modes in Module Configuration Register (FR_MCR)
- Configure the FIFO system memory base address
- Configure the Receive FIFO Start Index Register (FR_RFSIR) with the first message buffer header index that belongs to the FIFO
- Configure the Receive FIFO Start Data Offset Register (FR_RFSDOR) with the data field offset of the data field belonging to the first message buffer that belongs to the FIFO
- Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO entry size
- Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO depth
- Configure the FIFO Filters

50.7.9.3 FIFO Periodic Timer

The FIFO periodic timer is used to generate an FIFO almost-full interrupt at certain point in time, if the almost-full watermark is not reached, but the FIFO is not empty. This can be used to prevent frames from get stuck in the FIFO for a long time.

The FIFO periodic timer is configured via the Receive FIFO Periodic Timer Register (FR_RFPTR). If the periodic timer duration FR_RFPTR[PTD] is configured to 0x0000, the periodic timer is continuously expired. If the periodic timer duration FR_RFPTR[PTD] is configured to 0x3FFF, the periodic timer never expires. If the periodic timer is configured to a value *ptd*, greater than 0x0000 and smaller 0x3FFF, the periodic timer expires and is restarted at the start of every communication cycle, and expires and is restarted after *ptd* macroticks have been elapsed.

50.7.9.4 FIFO Reception

The FIFO reception is a CC internal operation.

A message frame reception is directed into the FIFO, if no individual message buffer is assigned for transmission or subscribed for reception for the current slot. In this case the FIFO filter path shown in Figure 50-21 is activated.

If the FIFO filter path indicates that the received frame has to be appended to the FIFO and the FIFO is not full, the CC writes the received frame header into the message buffer header field indicated by the CC internal FIFO write index. The frame payload data are written into the corresponding message buffer data field. If the status of the received frame indicates a valid non-null frame, the slot status information is written into the message buffer header field and the CC internal FIFO write index is updated by 1 and the fifo fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is incremented. If the status of the received frame indicates an invalid or null frame, the frame is not appended to the FIFO.

50.7.9.5 FIFO Almost-Full Interrupt Generation

If the fifo fill level FLA (FLB) is updated after a frame reception and exceeds the FIFO watermark level WM, i.e. $FLA>WM_A$ ($FLB>WM_B$), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

If the periodic timer expires, and FIFOA (FIFOB) is not empty, i.e. FLA>0 (FLB>0), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

50.7.9.6 FIFO Overflow Error Generation

If the FIFOA (FIFOB) is full, i.e. $FLA=FIFO\_DEPTH_A$ ($FLB=FIFO\_DEPTH_B$) and the conditions for a FIFO reception as described in FIFO Reception are fulfilled, then the fifo overflow error flag FR_CHIERFR[FOVA_EF] (FR_CHIERFR[FOVB_EF]) is asserted.

50.7.9.7 FIFO Message Access

The FIFOA (FIFOB) contains valid messages if the FIFO fill level given in the fields FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is greater than 0. The Receive FIFO A Read Index Register (FR_RFARIR) and the

(Receive FIFO B Read Index Register (FR_RFBRIR)) point to a message buffer with valid content and the oldest frames stored in the FIFO.The respective read data field offsets can be calculated according to Equation 1-125.

If the FIFO fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is 0, than the FIFOA (FIFOB) contains no valid messages and the corresponding read index register Receive FIFO A Read Index Register (FR_RFARIR) or (Receive FIFO B Read Index Register (FR_RFBRIR) point to a message buffer with invalid content. In this case the application must not read data from this FIFO.

To access the oldest message in the FIFOA (FIFOB), the application first reads the read index RDIDX out of the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)). This read index points to the message buffer header field of the oldest message buffer that contains valid received message data. The data field offset belonging to this message buffer must be calculated by the application according to Equation 1-125. The application can access the message data as described in Receive FIFO. When the application has read the message buffer data and status information, it can update the FIFO as described in FIFO Update.

50.7.9.8 FIFO Update

The application updates the FIFOA (FIFOB) by writing a pop count value *pc* different from 0 to the PCA (PCB) field in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR).

As a result of the this operation, the CC removes the oldest *pc* entries from FIFOA (FIFOB).

If the specified pop count value *pc* is greater than the current fill level *fl* provided in FLA (FAB) field, then only *fl* entries are removed from the FIFOA (FIFOB), the remaining *fl-pc* requested pop operations are discarded without any notification. In this case FIFOA (FIFOB) is empty after the update operation.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by the number of removed items. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index defined in Receive FIFO Start Index Register (FR_RFSIR) automatically.

50.7.9.8.1 FIFO Interrupt Flag Update

The FIFO Interrupt Flag Update mode is configured, when the FIFO update mode flag FR_MCR[FUM] is set to 0. In this mode FIFOA (FIFOB) will be updated by 1 entry, when the interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is written with 1 by the application.

If the FIFO is empty, the update request is ignored without any notification.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by 1, if the FIFO was not empty. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index automatically.

50.7.9.9 FIFO Filtering

The FIFO filtering is activated after all enabled individual receive message buffers have been searched without success for a message buffer to receive the current frame.

The CC provides three sets of FIFO filters. The FIFO filters are applied to valid non-null frames only. The FIFO will not receive invalid or null-frames. For each FIFO filter, the pass criteria is specified in the related section given below. Only frames that have passed all filters will be appended to the FIFO. The FIFO filter path is depicted in the figure below.

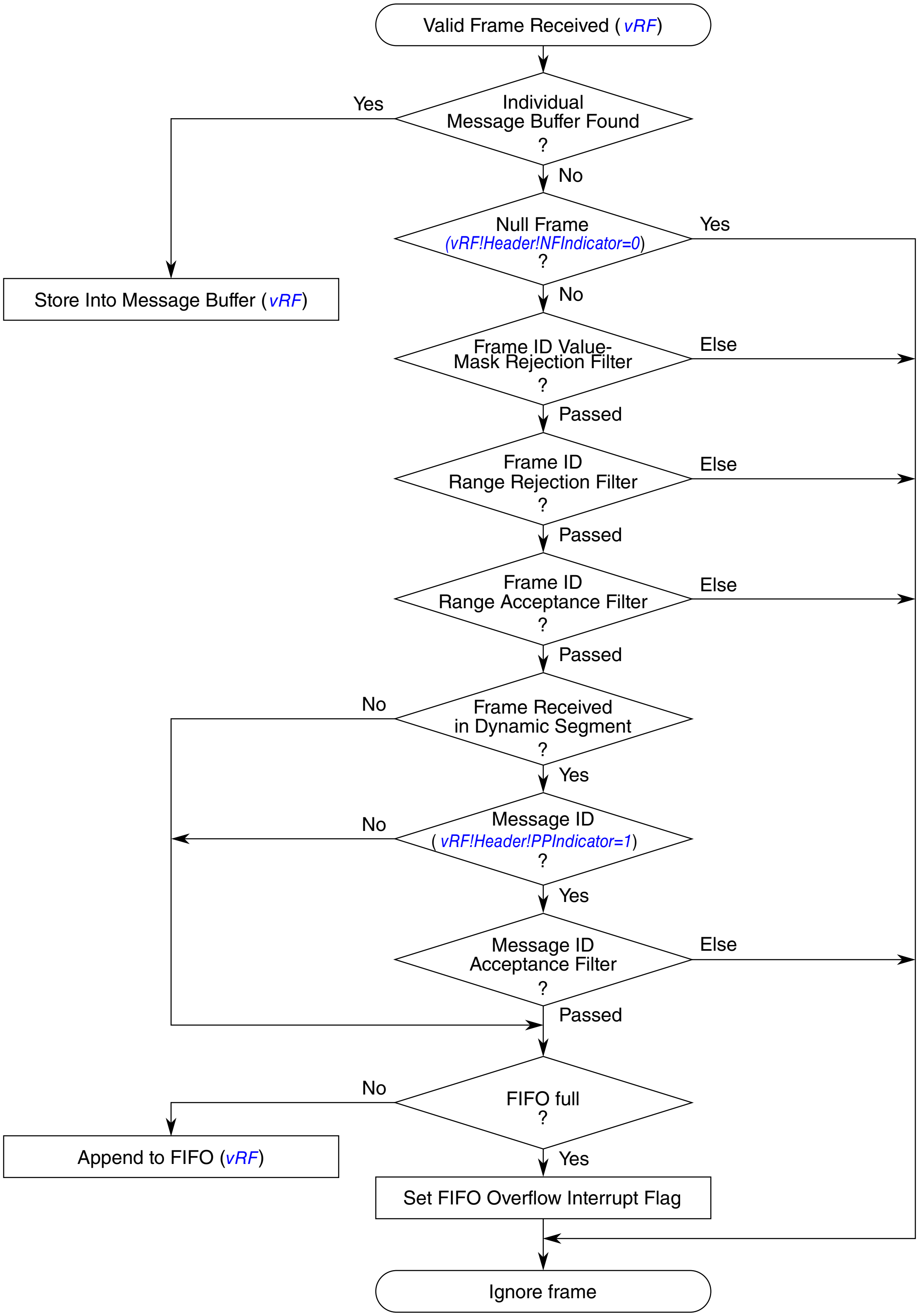

Valid Frame Received (vRF)
Individual Message Buffer Found ?
Yes
No
Store Into Message Buffer (vRF)
Null Frame (vRF!Header!NFIndicator=0) ?
Yes
No
Frame ID Value-Mask Rejection Filter ?
Else
Passed
Frame ID Range Rejection Filter ?
Else
Passed
Frame ID Range Acceptance Filter ?
Else
Passed
Frame Received in Dynamic Segment ?
No
Yes
Message ID (vRF!Header!PPIndicator=1) ?
No
Yes
Message ID Acceptance Filter ?
Else
Passed
FIFO full ?
No
Yes
Append to FIFO (vRF)
Set FIFO Overflow Interrupt Flag
Ignore frame

Figure 50-21. Received Frame FIFO Filter Path

A received frame passes the FIFO filtering if it has passed all three type of filter.

50.7.9.9.1 RX FIFO Frame ID Value-Mask Rejection Filter

The frame ID value-mask rejection filter is a value-mask filter and is defined by the fields in the Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR) and the Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR). Each received frame with a frame ID FID that does not match the value-mask filter value passes the filter, i.e. is not rejected.

Consequently, a received valid frame with the frame ID FID passes the RX FIFO Frame ID Value-Mask Rejection Filter if Equation 29 on page 1829 is fulfilled.

$$\mathrm{FID}\ \&\ \mathrm{FR\_RFFIDRFMR}[\mathrm{FIDRFMSK}] \neq \mathrm{FR\_RFFIDRFVR}[\mathrm{FIDRFVAL}]\ \&\ \mathrm{FR\_RFFIDRFMR}[\mathrm{FIDRFMSK}]$$

Equation 29

The RX FIFO Frame ID Value-Mask Rejection Filter can be configured to pass all frames by the following settings.

$$\mathrm{FR\_RFFIDRFVR}[\mathrm{FIDRFVAL}] = 0x000 \text{ and } \mathrm{FR\_RFFIDRFMR}[\mathrm{FIDRFMSK}] = 0\mathrm{x7FF}$$

Equation 30

Using the settings above, only the frame with frame ID 0 will be rejected, which is an invalid frame. All other frames will pass.

The RX FIFO Frame ID Value-Mask Rejection Filter can be configured to reject all frames by the following settings.

$$\mathrm{FR\_RFFIDRFMR}[\mathrm{FIDRFMSK}] = 0\mathrm{x}000$$

Equation 31

Using these settings, Equation 29 on page 1829 can never be fulfilled (0!= 0) and thus all frames are rejected; no frame will pass. This is the reset value for the RX FIFO.

50.7.9.9.2 RX FIFO Frame ID Range Rejection Filter

Each of the four RX FIFO Frame ID Range filters can be configured as a rejection filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range rejection filters if either no rejection filter is enabled, or, for all of the enabled RX FIFO Frame ID Range rejection filters, i.e. FR_RFRFCTR[FiMD] = 1 and FR_RFRFCTR[FiEN] = 1, Equation 32 on page 1830 is fulfilled.

$$FID < FR\_RFRFCFR_{SEL}[SID_{IBD=0}]) or (FR\_RFRFCFR_{SEL}[SID_{IBD=1}] < FID)$$

Equation 32

Consequently, all frames with a frame ID that fulfills Equation 33 on page 1830 for at least one of the enabled rejection filters will be rejected and thus not pass.

$$FR\_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR\_RFRFCFR_{SEL}[SID_{IBD=1}]$$

Equation 33

50.7.9.9.3 RX FIFO Frame ID Range Acceptance filter

Each of the four RX FIFO Frame ID Range filters can be configured as an acceptance filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range acceptance filters if either no acceptance filter is enabled, or, for at least one of the enabled RX FIFO Frame ID Range acceptance filters, i.e. FR_RFRFCTR[FiMD] = 0 and FR_RFRFCTR[FiEN] = 1, Equation 34 on page 1830 is fulfilled.

$$FR\_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR\_RFRFCFR_{SEL}[SID_{IBD=1}]$$

Equation 34

50.7.9.9.4 RX FIFO Message ID Acceptance Filter

The RX FIFO Message ID Acceptance Filter is a value-mask filter and is defined by the Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR) and the Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR). This filter applies only to valid frames received in the dynamic segment with the payload preamble indicator bit PPI set to 1. All other frames will pass this filter.

A received valid frame in the dynamic segment with the payload preamble indicator bit PPI set to 1 and with the message ID MID (the first two bytes of the payload) will pass the RX FIFO Message ID Acceptance Filter if Equation 35 on page 1830 is fulfilled.

$$MID \,\&\, FR\_RFMIDAFMR[MIDAFMSK] = FR\_RFMIDAFMR[MIDAFVAL] \,\&\, FR\_RFMIDAFMR[MIDAFMSK]$$

Equation 35

The RX FIFO Message ID Acceptance Filter can be configured to accept all frames by setting

$$FR\_\ RFMIDAFMR[MIDAFMSK]:=0x000$$

Equation 36

Using these settings, Equation 35 on page 1830 is always fulfilled and all frames will pass.

50.7.10 Channel Device Modes

This section describes the two FlexRay channel device modes that are supported by the CC.

50.7.10.1 Dual Channel Device Mode

In the dual channel device mode, both FlexRay ports are connected to physical FlexRay bus lines. The FlexRay port consisting of FR_A_RX, FR_A_TX, and $\overline{\text{FR\_A\_TX\_EN}}$ is connected to the physical bus channel A and the FlexRay port consisting of FR_B_RX, FR_B_TX, and $\overline{\text{FR\_B\_TX\_EN}}$ is connected to the physical bus channel B. The dual channel system is shown in the following figure.

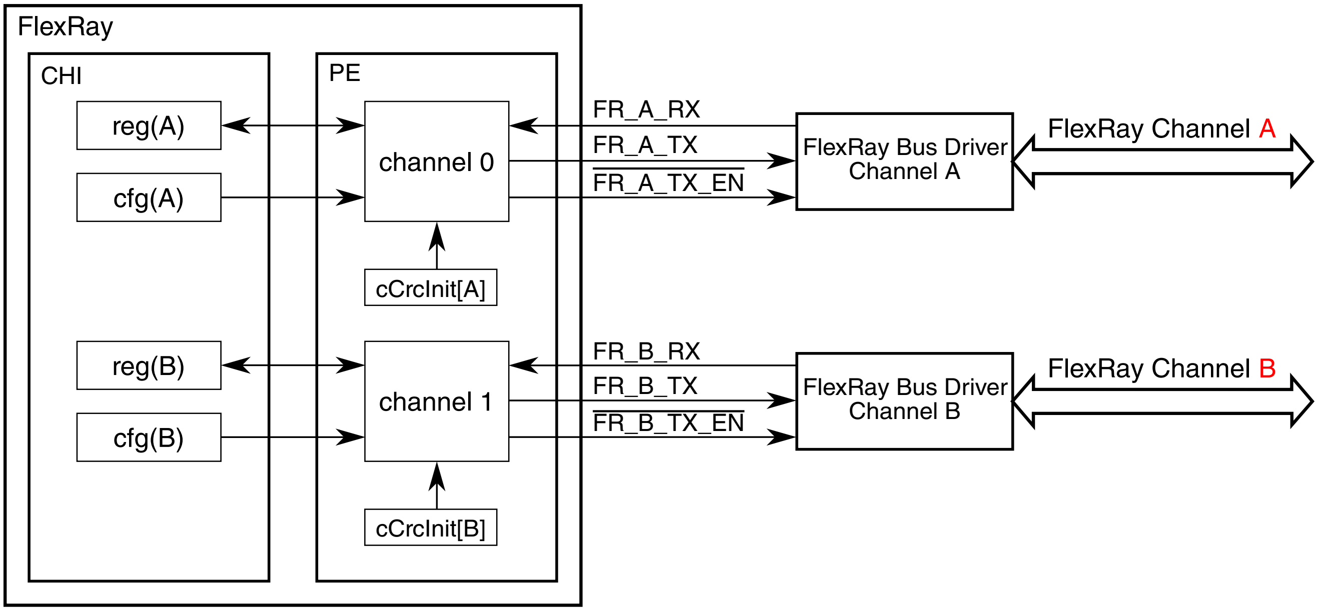


Figure 50-22. Dual Channel Device Mode

50.7.10.2 Single Channel Device Mode

The single channel device mode supports devices that have only one FlexRay port available. This FlexRay port consists of the signals FR_A_RX, FR_A_TX, and FR_A_TX_EN and can be connected to either the physical bus channel A (shown in the "Single Channel Device Mode (Channel A)" figure below) or the physical bus channel B (shown in the "Single Channel Device Mode (Channel B)" figure below).

If the device is configured as a single channel device by setting FR_MCR[SCM] to 1, only the internal channel A and the FlexRay Port A is used. Depending on the setting of FR_MCR[CHA] and FR_MCR[CHB], the internal channel A behaves either as a FlexRay Channel A or FlexRay Channel B. The bit FR_MCR[CHA] must be set, if the FlexRay Port A is connected to a FlexRay Channel A. The bit FR_MCR[CHB] must be set if the FlexRay Port A is connected to a FlexRay Channel B. The two FlexRay channels differ only in the initial value for the frame CRC *cCrcInit*. For a single channel device, the application can access and configure only the registers related to internal channel A.

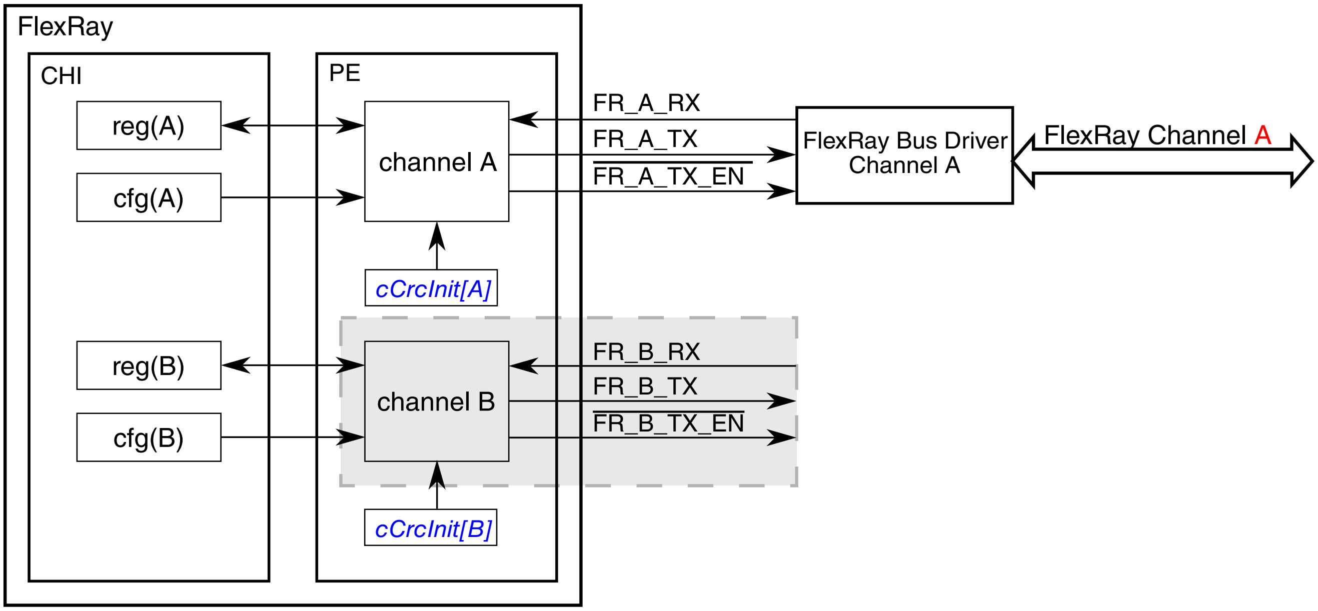


Figure 50-23. Single Channel Device Mode (Channel A)

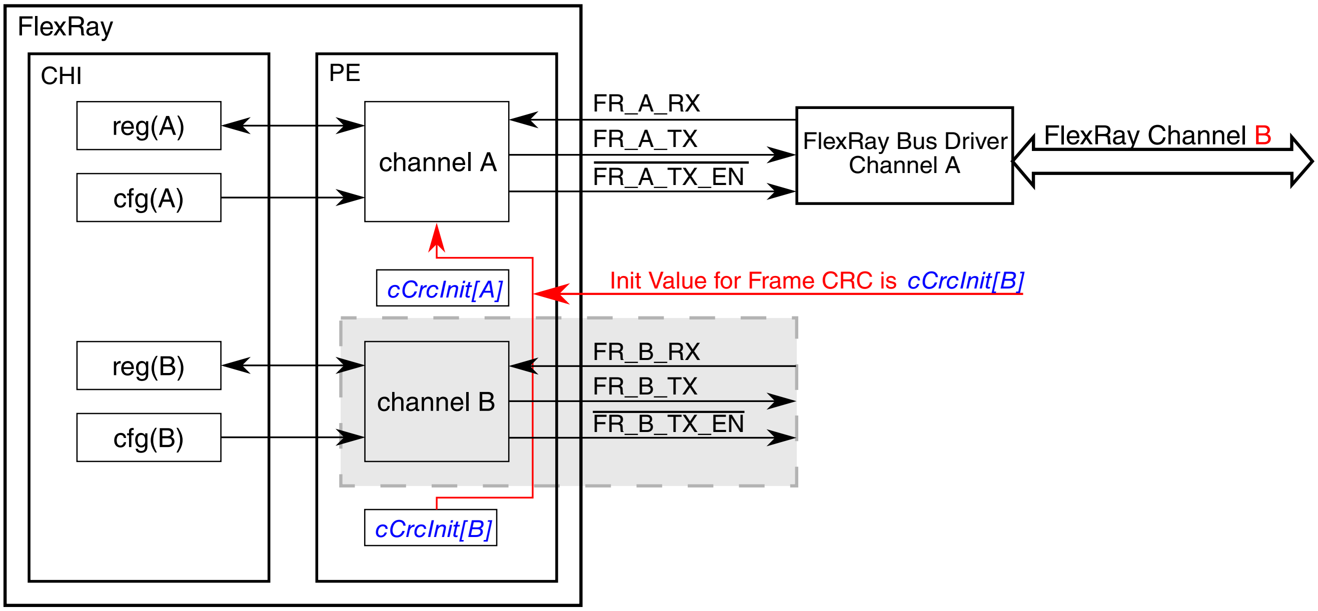


Figure 50-24. Single Channel Device Mode (Channel B)

50.7.11 External Clock Synchronization

The application of the external rate and offset correction is triggered when the application writes to the EOC_AP and ERC_AP fields in the Protocol Operation Control Register (FR_POCR). The PE applies the external correction values in the next even-odd cycle pair as shown in the "External Offset Correction Write and Application Timing" figure and the "External Rate Correction Write and Application Timing" figures below.

Note

> The values provided in the EOC_AP and ERC_AP fields are the values that were written from the application most recently. If these value were already applied, they will not be applied in the current cycle pair again.

If the offset correction applied in the NIT of cycle 2n+1 shall be affect by the external offset correction, the EOC_AP field must be written to after the start of cycle 2n and before the end of the static segment of cycle 2n+1. If this field is written to after the end of the static segment of cycle 2n+1, it is not guaranteed that the external correction value is applied in cycle 2n+1. If the value is not applied in cycle 2n+1, then the value will be applied in the cycle 2n+3. Refer to the following for timing details.

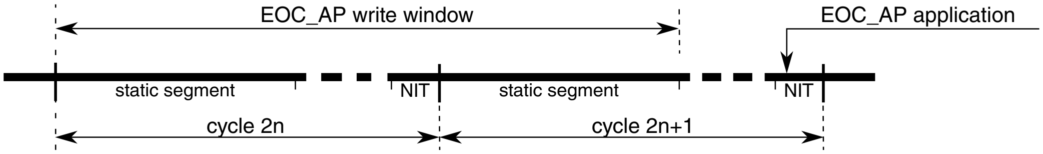


Figure 50-25. External Offset Correction Write and Application Timing

If the rate correction for the cycle pair [2n+2, 2n+3] shall be affect by the external offset correction, the ERC_AP field must be written to after the start of cycle 2n and before the end of the static segment start of cycle 2n+1. If this field is written to after the end of the static segment of cycle 2n+1, it is not guaranteed that the external correction value is applied in cycle pair [2n+2, 2n+3]. If the value is not applied for cycle pair [2n+2, 2n+3], then the value will be applied for cycle pair [2n+4, 2n+5]. Refer to the following for details.

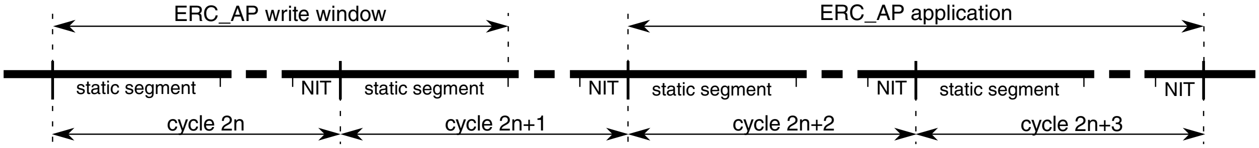


Figure 50-26. External Rate Correction Write and Application Timing

50.7.12 Sync Frame ID and Sync Frame Deviation Tables

The FlexRay protocol requires the provision of a snapshot of the Synchronization Frame ID tables for the even and odd communication cycle for both channels. The CC provides the means to write a copy of these internal tables into the FlexRay memory area and ensures application access to consistent tables by means of table locking. Once the application has locked the table successfully, the CC will not overwrite these tables and the application can read a consistent snapshot.

Note

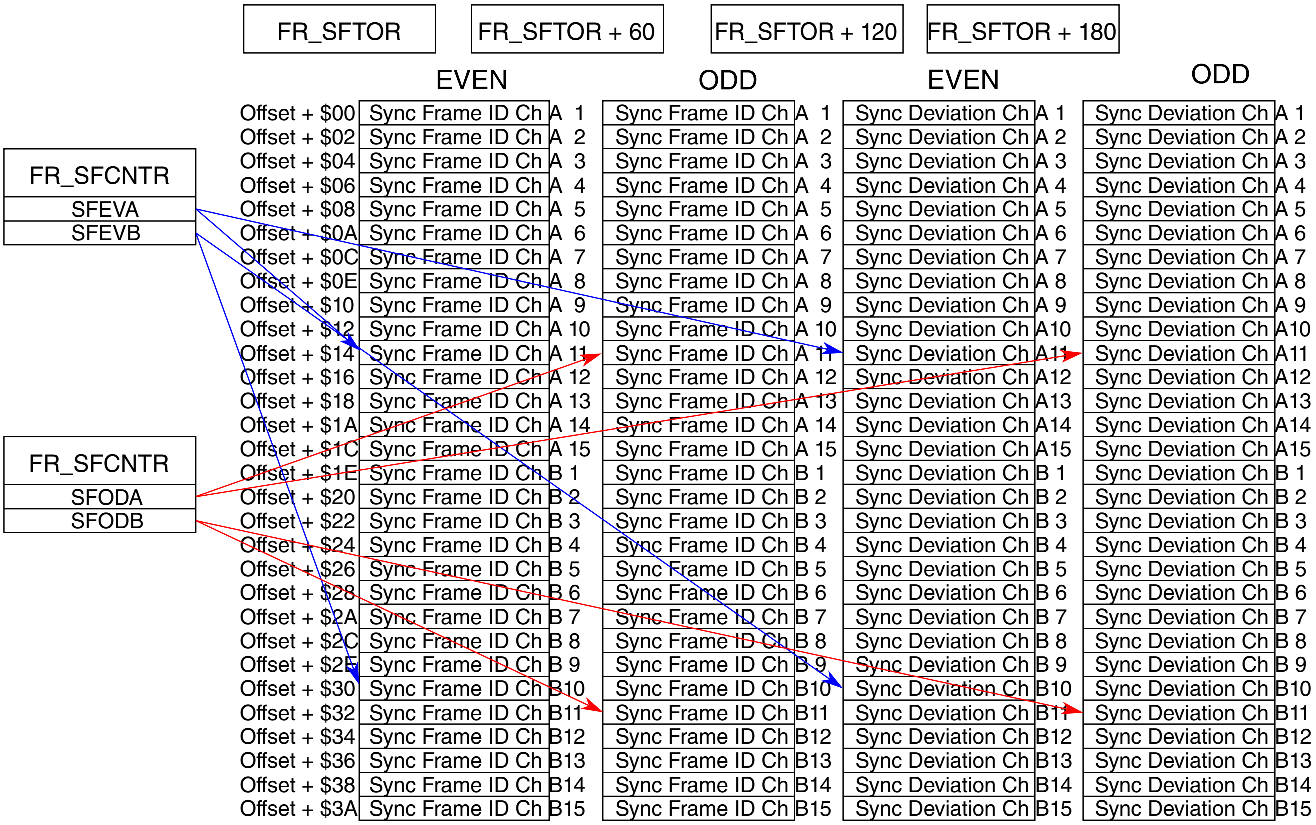
Only synchronization frames that have passed the synchronization frame filters are considered for clock synchronization and appear in the sync frame tables.

50.7.12.1 Sync Frame ID Table Content

The Sync Frame ID Table is a snapshot of the protocol related variables *vsSyncIdListA* and *vsSyncIdListB* for each even and odd communication cycle. This table provides a list of the frame IDs of the synchronization frames received on the corresponding channel and cycle that are used for the clock synchronization.

50.7.12.2 Sync Frame Deviation Table Content

The Sync Frame Deviation Table is a snapshot of the protocol related variable zsDev(id)(oe)(ch)!Value. Each Sync Frame Deviation Table entry provides the deviation value for the sync frame, with the frame ID presented in the corresponding entry in the Sync Frame ID Table.



	FR_SFTOR EVEN	FR_SFTOR + 60 ODD	FR_SFTOR + 120 EVEN	FR_SFTOR + 180 ODD
Offset + $00	Sync Frame ID Ch A 1	Sync Frame ID Ch A 1	Sync Deviation Ch A 1	Sync Deviation Ch A 1
Offset + $02	Sync Frame ID Ch A 2	Sync Frame ID Ch A 2	Sync Deviation Ch A 2	Sync Deviation Ch A 2
Offset + $04	Sync Frame ID Ch A 3	Sync Frame ID Ch A 3	Sync Deviation Ch A 3	Sync Deviation Ch A 3
Offset + $06	Sync Frame ID Ch A 4	Sync Frame ID Ch A 4	Sync Deviation Ch A 4	Sync Deviation Ch A 4
Offset + $08	Sync Frame ID Ch A 5	Sync Frame ID Ch A 5	Sync Deviation Ch A 5	Sync Deviation Ch A 5
Offset + $0A	Sync Frame ID Ch A 6	Sync Frame ID Ch A 6	Sync Deviation Ch A 6	Sync Deviation Ch A 6
Offset + $0C	Sync Frame ID Ch A 7	Sync Frame ID Ch A 7	Sync Deviation Ch A 7	Sync Deviation Ch A 7
Offset + $0E	Sync Frame ID Ch A 8	Sync Frame ID Ch A 8	Sync Deviation Ch A 8	Sync Deviation Ch A 8
Offset + $10	Sync Frame ID Ch A 9	Sync Frame ID Ch A 9	Sync Deviation Ch A 9	Sync Deviation Ch A 9
Offset + $12	Sync Frame ID Ch A 10	Sync Frame ID Ch A 10	Sync Deviation Ch A10	Sync Deviation Ch A10
Offset + $14	Sync Frame ID Ch A 11	Sync Frame ID Ch A 11	Sync Deviation Ch A11	Sync Deviation Ch A11
Offset + $16	Sync Frame ID Ch A 12	Sync Frame ID Ch A 12	Sync Deviation Ch A12	Sync Deviation Ch A12
Offset + $18	Sync Frame ID Ch A 13	Sync Frame ID Ch A 13	Sync Deviation Ch A13	Sync Deviation Ch A13
Offset + $1A	Sync Frame ID Ch A 14	Sync Frame ID Ch A 14	Sync Deviation Ch A14	Sync Deviation Ch A14
Offset + $1C	Sync Frame ID Ch A 15	Sync Frame ID Ch A 15	Sync Deviation Ch A15	Sync Deviation Ch A15
Offset + $1E	Sync Frame ID Ch B 1	Sync Frame ID Ch B 1	Sync Deviation Ch B 1	Sync Deviation Ch B 1
Offset + $20	Sync Frame ID Ch B 2	Sync Frame ID Ch B 2	Sync Deviation Ch B 2	Sync Deviation Ch B 2
Offset + $22	Sync Frame ID Ch B 3	Sync Frame ID Ch B 3	Sync Deviation Ch B 3	Sync Deviation Ch B 3
Offset + $24	Sync Frame ID Ch B 4	Sync Frame ID Ch B 4	Sync Deviation Ch B 4	Sync Deviation Ch B 4
Offset + $26	Sync Frame ID Ch B 5	Sync Frame ID Ch B 5	Sync Deviation Ch B 5	Sync Deviation Ch B 5
Offset + $28	Sync Frame ID Ch B 6	Sync Frame ID Ch B 6	Sync Deviation Ch B 6	Sync Deviation Ch B 6
Offset + $2A	Sync Frame ID Ch B 7	Sync Frame ID Ch B 7	Sync Deviation Ch B 7	Sync Deviation Ch B 7
Offset + $2C	Sync Frame ID Ch B 8	Sync Frame ID Ch B 8	Sync Deviation Ch B 8	Sync Deviation Ch B 8
Offset + $2E	Sync Frame ID Ch B 9	Sync Frame ID Ch B 9	Sync Deviation Ch B 9	Sync Deviation Ch B 9
Offset + $30	Sync Frame ID Ch B10	Sync Frame ID Ch B10	Sync Deviation Ch B10	Sync Deviation Ch B10
Offset + $32	Sync Frame ID Ch B11	Sync Frame ID Ch B11	Sync Deviation Ch B11	Sync Deviation Ch B11
Offset + $34	Sync Frame ID Ch B12	Sync Frame ID Ch B12	Sync Deviation Ch B12	Sync Deviation Ch B12
Offset + $36	Sync Frame ID Ch B13	Sync Frame ID Ch B13	Sync Deviation Ch B13	Sync Deviation Ch B13
Offset + $38	Sync Frame ID Ch B14	Sync Frame ID Ch B14	Sync Deviation Ch B14	Sync Deviation Ch B14
Offset + $3A	Sync Frame ID Ch B15	Sync Frame ID Ch B15	Sync Deviation Ch B15	Sync Deviation Ch B15

Figure 50-27. Sync Table Memory Layout

50.7.12.3 Sync Frame ID and Sync Frame Deviation Table Setup

The CC writes a copy of the internal synchronization frame ID and deviation tables into the FlexRay memory area if requested by the application. The application must provide the appropriate amount of FlexRay memory area for the tables. The memory layout of the tables is given in Figure 50-27. Each table occupies 120 16-bit entries.

While the protocol is in *POC:config* state, the application must program the offsets for the tables into the Sync Frame Table Offset Register (FR_SFTOR).

50.7.12.4 Sync Frame ID and Sync Frame Deviation Table Generation

The application controls the generation process of the Sync Frame ID and Sync Frame Deviation Tables into the FlexRay memory area using the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). A summary of the copy modes is given in the table below.

Table 50-52. Sync Frame Table Generation Modes

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	0	0	No Sync Frame Table copy
0	0	1	Sync Frame ID Tables will be copied continuously
0	1	0	Reserved
0	1	1	Sync Frame ID Tables and Sync Frame Deviation Tables will be copied continuously
1	0	0	No Sync Frame Table copy
1	0	1	Sync Frame ID Tables for next even-odd-cycle pair will be copied
1	1	0	Reserved
1	1	1	Sync Frame ID Tables and Sync Frame Deviation Tables for next even-odd-cycle pair will be copied

The Sync Frame Table generation process is described in the following for the even cycle. The same sequence applies to the odd cycle.

If the application has enabled the sync frame table generation by setting FR_SFTCCSR[SIDEN] to 1, the CC starts the update of the even cycle related tables after the start of the NIT of the next even cycle. The CC checks if the application has locked the tables by reading the FR_SFTCCSR[ELKS] lock status bit. If this bit is set, the CC will not update the table in this cycle. If this bit is cleared, the CC locks this table and starts the table update. To indicate that these tables are currently updated and may contain inconsistent data, the CC clears the even table valid status bit FR_SFTCCSR[EVAL]. Once all table entries related to the even cycle have been transferred into the FlexRay memory area, the CC sets the even table valid bit FR_SFTCCSR[EVAL] and the Even Cycle Table Written Interrupt Flag EVT_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). If the interrupt enable flag EVT_IE is set, an interrupt request is generated.

To read the generated tables, the application must lock the tables to prevent the CC from updating these tables. The locking is initiated by writing a 1 to the even table lock trigger FR_SFTCCSR[ELKT]. When the even table is not currently updated by the CC, the lock is granted and the even table lock status bit FR_SFTCCSR[ELKS] is set. This indicates that the application has successfully locked the even sync tables and the corresponding status information fields SFRA, SFRB in the Sync Frame Counter Register (FR_SFCNTR). The value in the FR_SFTCCSR[CYCNUM] field provides the number of the cycle that this table is related to.

The number of available table entries per channel is provided in the FR_SFCNTR[SFEVA] and FR_SFCNTR[SFEVB] fields. The application can now start to read the sync table data from the locations given in Figure 50-27.

After reading all the data from the locked tables, the application must unlock the table by writing to the even table lock trigger FR_SFTCCSR[ELKT] again. The even table lock status bit FR_SFTCCSR[ELKS] is reset immediately.

If the sync frame table generation is disabled, the table valid bits FR_SFTCCSR[EVAL] and FR_SFTCCSR[EVAL] are reset when the counter values in the Sync Frame Counter Register (FR_SFCNTR) are updated. This is done because the tables stored in the FlexRay memory area are no longer related to the values in the Sync Frame Counter Register (FR_SFCNTR).

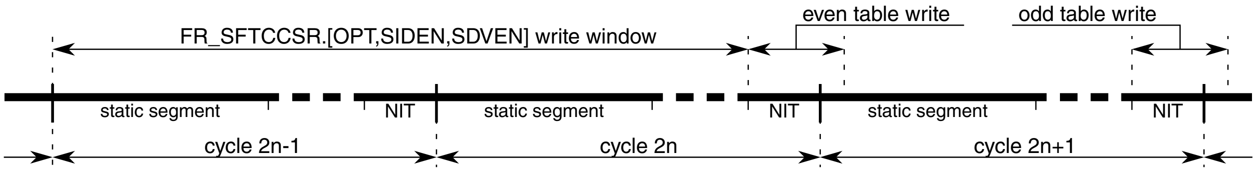


Figure 50-28. Sync Frame Table Trigger and Generation Timing

50.7.12.5 Sync Frame Table Access

The sync frame tables will be transferred into the FlexRay memory area during the table write windows shown in Figure 50-27. During the table write, the application can not lock the table that is currently written. If the application locks the table outside of the table write window, the lock is granted immediately.

50.7.12.5.1 Sync Frame Table Locking and Unlocking

The application locks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT in the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). If the affected table is not currently written to the FlexRay memory area, the lock is granted immediately, and the lock status bit ELKS/OLKS is set. If the affected table is currently written to the FlexRay memory area, the lock is not granted. In this case, the application must issue the lock request again until the lock is granted.

The application unlocks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT. The lock status bit ELKS/OLKS is cleared immediately.

50.7.13 MTS Generation

The CC provides a flexible means to request the transmission of the Media Access Test Symbol MTS in the symbol window on channel A or channel B.

The application can configure the set of communication cycles in which the MTS will be transmitted over the FlexRay bus by programming the CYCCNTMSK and CYCCNTVAL fields in the MTS A Configuration Register (FR_MTSACFR) and MTS B Configuration Register (FR_MTSBCFR).

The application enables or disables the generation of the MTS on either channel by setting or clearing the MTE control bit in the MTS A Configuration Register (FR_MTSACFR) or MTS B Configuration Register (FR_MTSBCFR). If an MTS is to be transmitted in a certain communication cycle, the application must set the MTE control bit during the static segment of the preceding communication cycle.

The MTS is transmitted over channel A in the communication cycle with number CYCCNT, if the following equations are fulfilled.

$$\mathrm{FR\_PSR0}[\mathrm{PROTSTATE}] = \mathrm{POC{:}\ normalactive}$$

Equation 37

$$\mathrm{FR\_MTSACRF}[\mathrm{MTE}] = 1$$

Equation 38

$$\mathrm{CYCCNT} \,\&\, \mathrm{FR\_MTSACFR}[\mathrm{CYCCNTMSK}] = \mathrm{FR\_MTSACFR}[\mathrm{CYCCNTVAL}] \,\&\, \mathrm{FR\_MTSACFR}[\mathrm{CYCCNTMSK}]$$

Equation 39

The MTS is transmitted over channel B in the communication cycle with number CYCCNT, if the following equations are fulfilled.

$$\mathrm{FR\_MTSBCRF}[\mathrm{MTE}] = 1$$

Equation 40

$$\mathrm{CYCCNT} \,\&\, \mathrm{FR\_MTSBCFR}[\mathrm{CYCCNTMSK}] = \mathrm{FR\_MTSBCFR}[\mathrm{CYCCNTVAL}] \,\&\, \mathrm{FR\_MTSBCFR}[\mathrm{CYCCNTMSK}]$$

Equation 41

50.7.14 Key Slot Transmission

Key slot assignment, transmission in POC:startup state, and transmission in POC:normal active state are discussed in this section.

50.7.14.1 Key Slot Assignment

A key slot is assigned to the CC if the key_slot_id field in the Protocol Configuration Register 18 (FR_PCR18) is configured with a value greater than 0 and less or equal to number_of_static_slots in Protocol Configuration Register 2 (FR_PCR2), otherwise no key slot is assigned.

50.7.14.2 Key Slot Transmission in POC:startup

If a key slot is assigned and the CC is in the *POC:startup* state, startup null frames will be transmitted as specified by FlexRay Communications System Protocol Specification, Version *2.1 Rev A* .

50.7.14.3 Key Slot Transmission in POC:normal active

If a key slot is assigned and the CC is in *POC:normal active*, a frame of the type as shown in the following table is transmitted. If a transmit message buffer is configured for the key slot and a valid message is available, a message frame is transmitted (see Message Transmission). If no transmit message buffer is configured for the key slot or no valid message is available, a null frame is transmitted (see Null Frame Transmission).

Table 50-53. Key Slot Frame Type

FR_PCR11[key_slot_used_for_sync]	FR_PCR11[key_slot_used_for_startup]	key slot frame type
0	0	normal frame
0	1	normal frame1[1]
1	0	sync frame
1	1	startup frame

[1] The frame transmitted has an semantically incorrect header and will be detected as an invalid frame at the receiver.

50.7.15 Sync Frame Filtering

Each received synchronization frame must pass the Sync Frame Acceptance Filter and the Sync Frame Rejection Filter before it is considered for clock synchronization. If the synchronization frame filtering is globally disabled, i.e. the SFFE control bit in the Module Configuration Register (FR_MCR) is cleared, all received synchronization frames are considered for clock synchronization. If a received synchronization frame did not pass at least one of the two filters, this frame is processed as a normal frame and is not considered for clock synchronization.

50.7.15.1 Sync Frame Acceptance Filtering

The synchronization frame acceptance filter is implemented as a value-mask filter. The value is configured in the Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR) and the mask is configured in the Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR). A received synchronization frame with the frame ID FID passes the sync frame acceptance filter, if the following equations evaluates to true.

$$\mathrm{FR\_MCR}[\mathrm{SFFE}] = 0$$

Equation 42

$$\mathrm{FID}\ \&\ \mathrm{FR\_SFIDAFMR}[\mathrm{FMSK}] = \mathrm{FR\_SFIDAFVR}[\mathrm{FVAL}]\ \&\ \mathrm{FR\_SFIDAFMR}[\mathrm{FMSK}]$$

Equation 43

Note

Sync frames are transmitted in the static segment only. Thus FID <= 1023.

50.7.15.2 Sync Frame Rejection Filtering

The synchronization frame rejection filter is a comparator. The compare value is defined by the Sync Frame ID Rejection Filter Register (FR_SFIDRFR). A received synchronization frame with the frame ID FID passes the sync frame rejection filter if the following equations evaluates to true.

$$\mathrm{FR\_MCR}[\mathrm{SFFE}] = 0$$

Equation 44

$$\mathrm{FID} \neq \mathrm{FR\_SFIDRFR}[\mathrm{SYNFRID}]$$

Equation 45

Note

Sync frames are transmitted in the static segment only. Thus FID <= 1023.

50.7.16 Strobe Signal Support

The CC provides a number of strobe signals for observing internal protocol timing related signals in the protocol engine. The signals are listed and described in the Strobe Signal Mapping table, located in the Register Descriptions section.

50.7.16.1 Strobe Signal Assignment

Each of the strobe signals listed in the Strobe Signal Mapping table, located in the Register Descriptions section, can be assigned to one of the four strobe ports using the Strobe Signal Control Register (FR_STBSCR). To assign multiple strobe signals, the application must write multiple times to the Strobe Signal Control Register (FR_STBSCR) with appropriate settings.

To read out the current settings for a strobe signal with number N, the application must execute the following sequence.

1. Write to FR_STBSCR with WMD = 1 and SEL = N. (updates SEL field only)
2. Read STBCSR.

   The SEL field provides N and the ENB and STBPSEL fields provides the settings for signal N.

50.7.16.2 Strobe Signal Timing

This section provides detailed timing information of the strobe signals with respect to the protocol engine clock.

The strobe signals display internal PE signals. Due to the internal architecture of the PE, some signals are generated several PE clock cycles before the actual action is performed on the FlexRay Bus. These signals are listed in the Strobe Signal Mapping table, located in the Register Descriptions section, with a negative clock offset. An example waveform is given in the following figure.

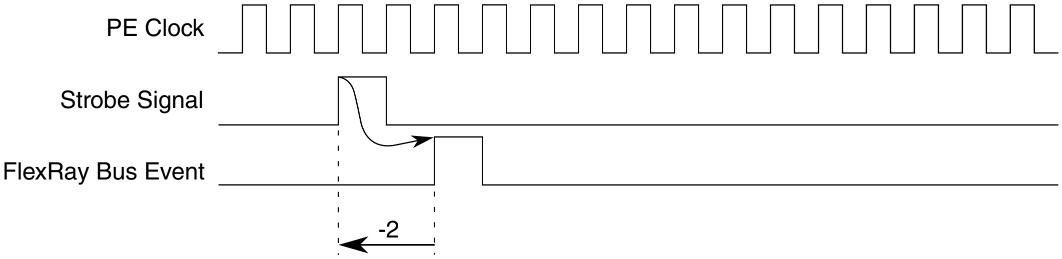


Figure 50-29. Strobe Signal Timing (type = pulse, clk_offset = -2)

Other signals refer to events that occurred on the FlexRay Bus some cycles before the strobe signal is changed. These signals are listed in the Strobe Signal Mapping table, located in the Register Descriptions section, with a positive clock offset. An example waveform is given in the figure below.

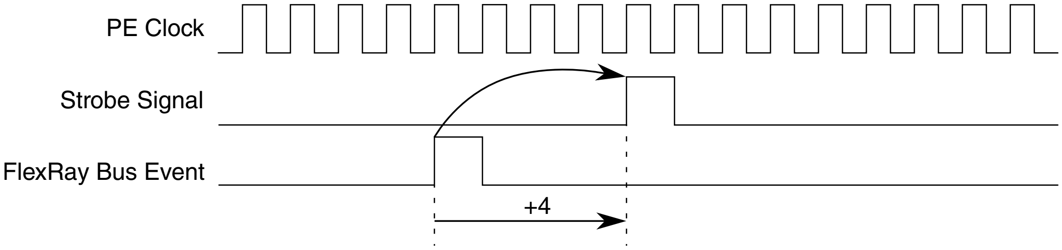


Figure 50-30. Strobe Signal Timing (type = pulse, clk_offset = +4)

50.7.17 Timer Support

The CC provides two timers, which run on the FlexRay time base. Each timer generates a maskable interrupt when it reaches a configured point in time. Timer T1 is an absolute timer. Timer T2 can be configured to be an absolute or a relative timer. Both timers can be configured to be repetitive. In the non-repetitive mode, timer stops if it expires. In repetitive mode, timer is restarted when it expires.

Both timers are active only when the protocol is in *POC:normal active* or *POC:normal passive* state. If the protocol is not in one of these modes, the timers are stopped. The application must restart the timers when the protocol has reached the *POC:normal active* or *POC:normal passive* state.

50.7.17.1 Absolute Timer T1

The absolute timer T1 has the protocol cycle count and the macrotick count as the time base. The timer 1 interrupt flag TI1_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set at the macrotick start event, if the following equations are fulfilled.

$$\mathrm{CYCTR[CTCCNT]\ \&\ FR\_TI1CYSR[}T\mathrm{1\_CYC\_MSK] = FR\_TI1CYSR[}T\mathrm{1\_CYC\_VAL]\ \&\ FR\_TI1CYSR[}T\mathrm{1\_CYC\_MSK]}$$

Equation 46

$$\mathrm{FR\_MTCTR[MTCT] = FR\_TI1MTOR[}T\mathrm{1\_MTOFFSET]}$$

Equation 47

If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The status bit T1ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T1ST bit is not cleared and the timer is restarted immediately. The T1ST is cleared when the timer is stopped.

The corresponding Interrupt condition (i.e., when the abovementioned equations are fulfilled) also leads to an event out indication on an output port (*fr_evt_tim1*) based on the [TIM1_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

50.7.17.2 Absolute / Relative Timer T2

The timer T2 can be configured to be an absolute or relative timer by setting the T2_CFG control bit in the Timer Configuration and Control Register (FR_TICCR). The status bit T2ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T2ST bit is not cleared and the timer is restarted immediately. The T2ST is cleared when the timer is stopped.

50.7.17.2.1 Absolute Timer T2

If timer T2 is configured as an absolute timer, it has the same functionality timer T1 but the configuration from Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1) is used. On expiration of timer T2, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set. If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The corresponding Interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the [TIM2_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

50.7.17.2.2 Relative Timer T2

If the timer T2 is configured as a relative timer, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set, when the programmed amount of macroticks MT[31:0], defined by Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1), has expired since the trigger or restart of timer 2. The relative timer is implemented as a down counter and expires when it has reached 0. At the macrotick start event, the value of MT[31:0] is checked and then decremented. Thus, if the timer is started with MT[31:0] == 0, it expires at the next macrotick start.

The corresponding Interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the [TIM2_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

50.7.18 Slot Status Monitoring

The CC provides several means for slot status monitoring. All slot status monitors use the same slot status vector provided by the PE. The PE provides a slot status vector for each static slot, for each dynamic slot, for the symbol window, and for the NIT, on a per channel base. The content of the slot status vector is described in the "Slot Status Content" table below. The PE provides the slot status vector within the first macrotick after the end of the related slot/window/NIT, as shown in the below figure.

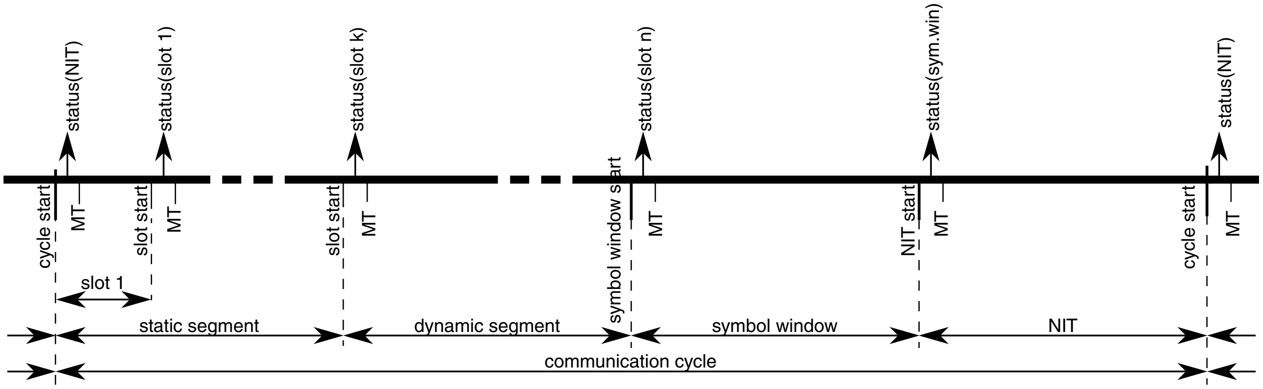


Figure 50-31. Slot Status Vector Update

Note

The slot status for the NIT of cycle n is provided after the start of cycle n+1.

Table 50-54. Slot Status Content

	Status Content
static /dynamic Slot	**slot related status**

Table continues on the next page...

Table 50-54. Slot Status Content (continued)

	Status Content
	vSS!ValidFrame - valid frame received *vSS!SyntaxError* - syntax error occurred while receiving *vSS!ContentError* - content error occurred while receiving *vSS!BViolation* - boundary violation while receiving *for slots in which the module transmits:* *vSS!TxConflict* - reception ongoing while transmission starts *for slots in which the module does not transmit:* *vSS!TxConflict* - reception ongoing while transmission starts first valid - channel that has received the first valid frame **received frame related status** *extracted from* *a)header of valid frame, if vSS!ValidFrame = 1* *b) last received header, if vSS!ValidFrame = 0* *c) set to 0, if nothing was received* *vRF!Header!NFIndicator* - Null Frame Indicator (0 for null frame) *vRF!Header!SuFIndicator* - Startup Frame Indicator *vRF!Header!SyFIndicator* - Sync Frame Indicator
Symbol Window	**window related status** *vSS!ValidFrame* - always 0 *vSS!ContentError* - content error occurred while receiving *vSS!SyntaxError* - syntax error occurred while receiving *vSS!BViolation* - boundary violation while receiving *vSS!TxConflict* - reception ongoing while transmission starts **received symbol related status** *vSS!ValidMTS* - valid Media Test Access Symbol received **received frame related status** *see static/dynamic slot*
NIT	**NIT related status** *vSS!ValidFrame* - always 0 *vSS!ContentError* - content error occurred while receiving *vSS!SyntaxError* - syntax error occurred while receiving *vSS!BViolation* - boundary violation while receiving *vSS!TxConflict* - always 0 **received frame related status** *see static/dynamic slot*

50.7.18.1 Channel Status Error Counter Registers

The two channel status error counter registers, Channel A Status Error Counter Register (FR_CASERCR) and Channel B Status Error Counter Register (FR_CBSERCR), incremented by one, if at least one of four slot status error bits, *vSS!SyntaxError, vSS!ContentError, vSS!BViolation*, or *vSS!TxConflict* is set to 1. The status vectors for all slots in the static and dynamic segment, in the symbol window, and in the NIT are taken into account. The counters wrap round after they have reached the maximum value.

50.7.18.2 Protocol Status Registers

The Protocol Status Register 2 (FR_PSR2) provides slot status information about the Network Idle Time NIT and the Symbol Window. The Protocol Status Register 3 (FR_PSR3) provides aggregated slot status information.

50.7.18.3 Slot Status Registers

The eight slot status registers, Slot Status Registers (FR_SSR0–FR_SSR7), can be used to observe the status of static slots, dynamic slots, the symbol window, or the NIT without individual message buffers. These registers provide all slot status related and received frame / symbol related status information, as given in Table 50-54, except of the *first valid* indicator for non-transmission slots.

50.7.18.4 Slot Status Counter Registers

The CC provides four slot status error counter registers, Slot Status Counter Registers (FR_SSCR0–FR_SSCR3). Each of these slot status counter registers is updated with the value of an internal slot status counter at the start of a communication cycle. The internal slot status counter is incremented if its increment condition, defined by the Slot Status Counter Condition Register (FR_SSCCR), matches the status vector provided by the PE. All static slots, the symbol window, and the NIT status are taken into account. *Dynamic* slots are *excluded*. The internal slot status counting and update timing is shown in the figure below.

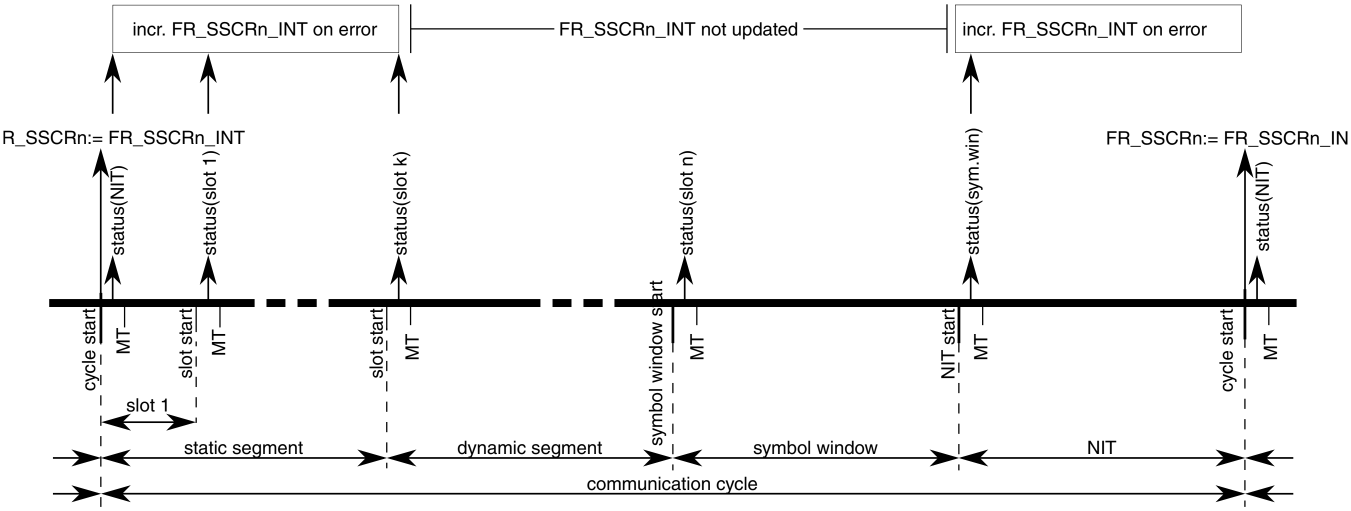


Figure 50-32. Slot Status Counting and FR_SSCRn Update

The PE provides the status of the NIT in the first slot of the next cycle. Due to these facts, the FR_SSCRn register reflects, in cycle n, the status of the NIT of cycle n-2, and the status of all static slots and the symbol window of cycle n-1.

The increment condition for each slot status counter consists of two parts, the frame related condition part and the slot related condition part. The internal slot status counter FR_SSCRn_INT is incremented if at least one of the conditions is fulfilled:

1. frame related condition:

   (FR_SSCCRn[VFR] | FR_SSCCRn[SYF] | FR_SSCCRn[NUF] | FR_SSCCRn[SUF]) // count on frame condition

   = 1;

   and

   ((~FR_SSCCRn[VFR] | vSS!ValidFrame) & // valid frame restriction

   (~FR_SSCCRn[SYF] | vRF!Header!SyFIndicator) & // sync frame indicator restriction

   (~FR_SSCCRn[NUF] | ~vRF!Header!NFIndicator) & // null frame indicator restriction

   (~FR_SSCCRn[SUF] | vRF!Header!SuFIndicator)) // startup frame indicator restriction

   = 1;

Note

> The indicator bits SYF, NUF, and SUF are valid only when a valid frame was received. Thus it is required to set the VFR always, whenever count on frame condition is used.

slot related condition:

((FR_SSCCRn[STATUSMASK[3]] & vSS!ContentError) | // increment on content error

(FR_SSCCRn[STATUSMASK[2]] & vSS!SyntaxError) | // increment on syntax error

(FR_SSCCRn[STATUSMASK[1]] & vSS!BViolation) | // increment on boundary violation

(FR_SSCCRn[STATUSMASK[0]] & vSS!TxConflict)) // increment on transmission conflict

= 1;

If the slot status counter is in single cycle mode, i.e. FR_SSCCRn[MCY] = 0, the internal slot status counter FR_SSCRn_INT is reset at each cycle start. If the slot status counter is in the multicycle mode, i.e. FR_SSCCRn[MCY] = 1, the counter is not reset and incremented, until the maximum value is reached.

50.7.18.5 Message Buffer Slot Status Field

Each individual message buffer and each FIFO message buffer provides a slot status field, which provides the information shown in Table 50-54 for the static/dynamic slot. The update conditions for the slot status field depend on the message buffer type. Refer to the Message Buffer Update Sections in Individual Message Buffer Configuration Data.

50.7.19 System Bus Access

This section provides a description of the system bus accesses failures and the related CC behavior. System bus access failures may occur when the CC transfers data to or from the FlexRay memory area.

The system bus access failure types are described in System Bus Access Failure Types.

The behavior of the CC after the occurrence of a system bus access failure is described in System Bus Access Failure Response.

50.7.19.1 System Bus Access Failure Types

This section describes the two types of system bus access failures.

50.7.19.1.1 System Bus Illegal Address Access

A system bus illegal address access is detected when the CC has used an illegal or invalid address to access the FlexRay system memory area. There are three conditions which are treated as a system bus illegal address access:

- The system bus subsystem detects an CC access to an illegal system memory address.
- The CC detects the usage of an data field offset with the value of 0.
- The CC detects a memory error while reading a data field offset from the CHI LRAM memory (see CHI LRAM Error Response after CC Read).

If a system bus illegal address access is detected, the CC sets the ILSA_EF flag in the CHI Error Flag Register (FR_CHIERFR).

50.7.19.1.2 System Bus Access Timeout

A system bus access timout is detected if an access to the FlexRay memory area is not finished in time. The timeout value is derived from the SYMATOR[TIMEOUT] setting (see Configure System Memory Access Time-Out Register (FR_SYMATOR)).

If a system bus access timout is detected, the CC sets the SBCF_EF flag in the CHI Error Flag Register (FR_CHIERFR).

50.7.19.2 System Bus Access Failure Response

This section describes the two types of behavior of the CC after the occurrence of a system bus access failure. The actual behavior is defined by the SBFF bit in the Module Configuration Register (FR_MCR).

50.7.19.2.1 Continue after System Bus Access Failure

If the SBFF bit in the Module Configuration Register (FR_MCR) is 0, the CC will continue its operation after the occurrence of the system bus access failure, but will not generate any system bus accesses until the start of the next communication cycle.Since no data are read from or written to the FlexRay memory area, no messages are received or transmitted. Consequently, none of the individual message buffers or receive FIFOs will be updated until the next communication cycle starts.

If a frame is under transmission when the system bus failure occurs, a correct frame is generated with the remaining header and frame data are replaced by all zeros. Depending on the point in time this can affect the PPI bit, the Header CRC, the Payload Length in case of an dynamic slot, and the payload data. Starting from the next slot in the current cycle, no frames will be transmitted and received, except for the key slot, where a sync or startup null-frame is transmitted, if the key slot is assigned.

If a frame is received when the system bus failure occurs, the reception is aborted and the related receive message buffer is not updated.

Normal operation is resumed after the start of next communication cycle.

50.7.19.2.2 Freeze after System Bus Access Failure

If the SBFF bit in the Module Configuration Register (FR_MCR) is set to 1, the CC will go into the freeze mode immediately after the occurrence of one of the system bus access failures.

50.7.20 Interrupt Support

The CC provides 108 individual interrupt sources and five combined interrupt sources.

50.7.20.1 Individual Interrupt Sources

Message buffer interrupts are discussed in this section.

50.7.20.1.1 Message Buffer Interrupts

The CC provides 64 message buffer interrupt sources.

Each individual message buffer provides an interrupt flag FR_MBCCSRn[MBIF] and an interrupt enable bit FR_MBCCSRn[MBIE]. The CC sets the interrupt flag when the slot status of the message buffer was updated. If the interrupt enable bit is asserted, an interrupt request is generated.

50.7.20.1.2 FIFO Interrupts

The CC provides 2 FIFO interrupt sources.

Each of the 2 FIFO provides a Receive FIFO Almost Full Interrupt Flag. The CC sets the Receive FIFO Almost Full Interrupt Flags (FR_GIFER[FAFBIF], FR_GIFER[FAFAIF]) in the Global Interrupt Flag and Enable Register (FR_GIFER) if the corresponding Receive FIFO fill level exceeds the defined watermark.

50.7.20.1.3 Wakeup Interrupt

The CC provides one interrupt source related to the wakeup.

The CC sets the Wakeup Interrupt Flag FR_GIFER[WUPIF] when it has received a wakeup symbol on the FlexRay bus. The CC generates an interrupt request if the interrupt enable bit FR_GIFER[WUPIE] is asserted.

50.7.20.1.4 Protocol Interrupts

The CC provides 25 interrupt sources for protocol related events. For details, see Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1). Each interrupt source has its own interrupt enable bit.

50.7.20.1.5 CHI Interrupts

The CC provides 16 interrupt sources for CHI related error events. For details, see CHI Error Flag Register (FR_CHIERFR). There is one common interrupt enable bit FR_GIFER[CHIE] for all CHI error interrupt sources.

50.7.20.2 Combined Interrupt Sources

Each combined interrupt source generates an interrupt request only when at least one of the interrupt sources that is combined generates an interrupt request.

50.7.20.2.1 Receive Message Buffer Interrupt

The Receive Message Buffer Interrupt request is generated when at least one of the individual receive message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[RBIE] is set.

50.7.20.2.2 Transmit Message Buffer Interrupt

The Transmit Message Buffer Interrupt request is generated when at least one of the individual transmit message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[TBIE] is asserted.

50.7.20.2.3 Protocol Interrupt

The Protocol Interrupt request is generated when at least one of the individual protocol interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[PRIE] is set..

50.7.20.2.4 CHI Interrupt

The CHI Interrupt request is generated when at least one of the individual chi error interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[CHIE] is set.

50.7.20.2.5 Module Interrupt

The Module Interrupt request is generated if at least one of the combined interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[MIE] is set.

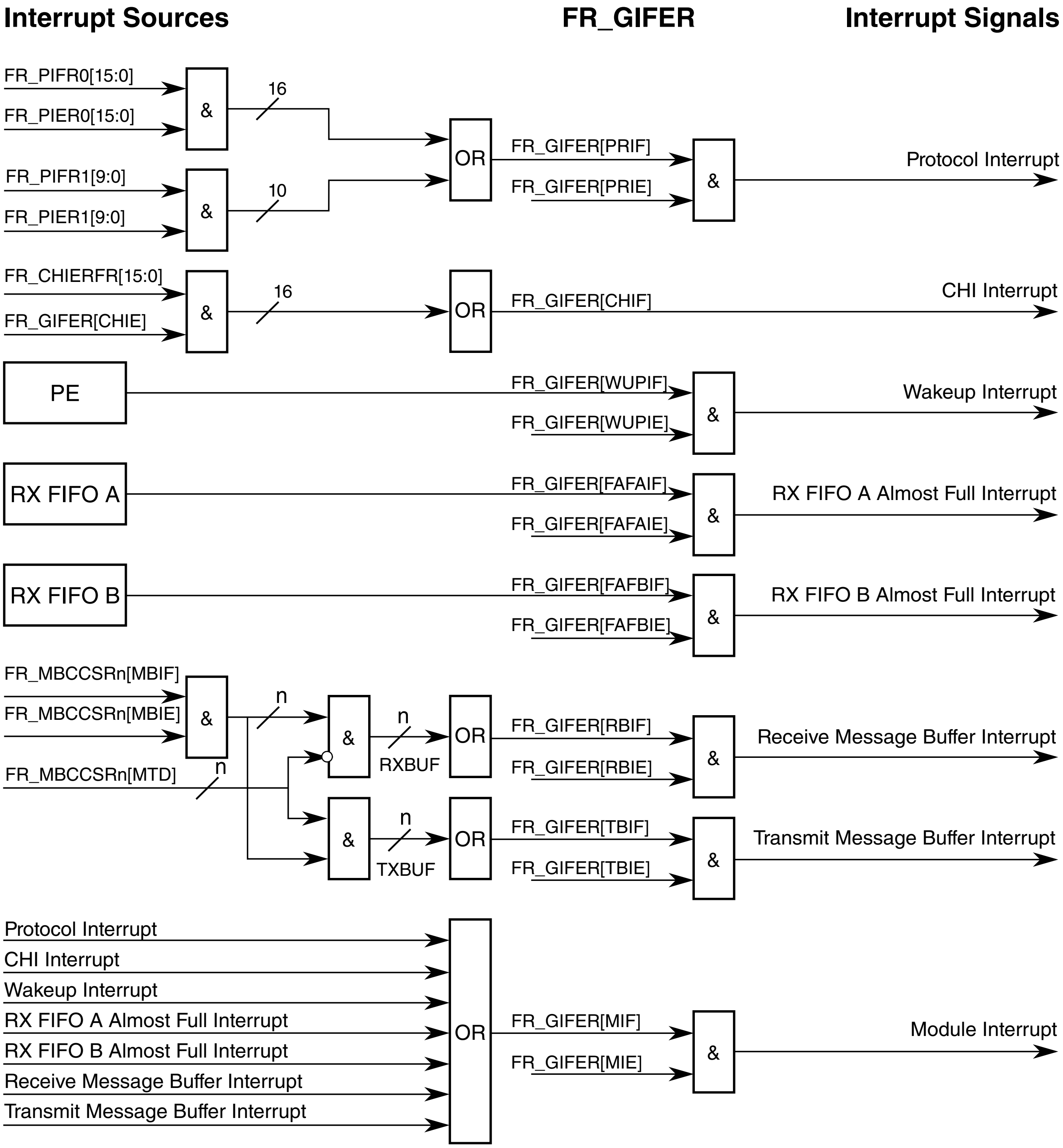


Figure 50-33. Scheme of FR_GIFER interrupt signal generation

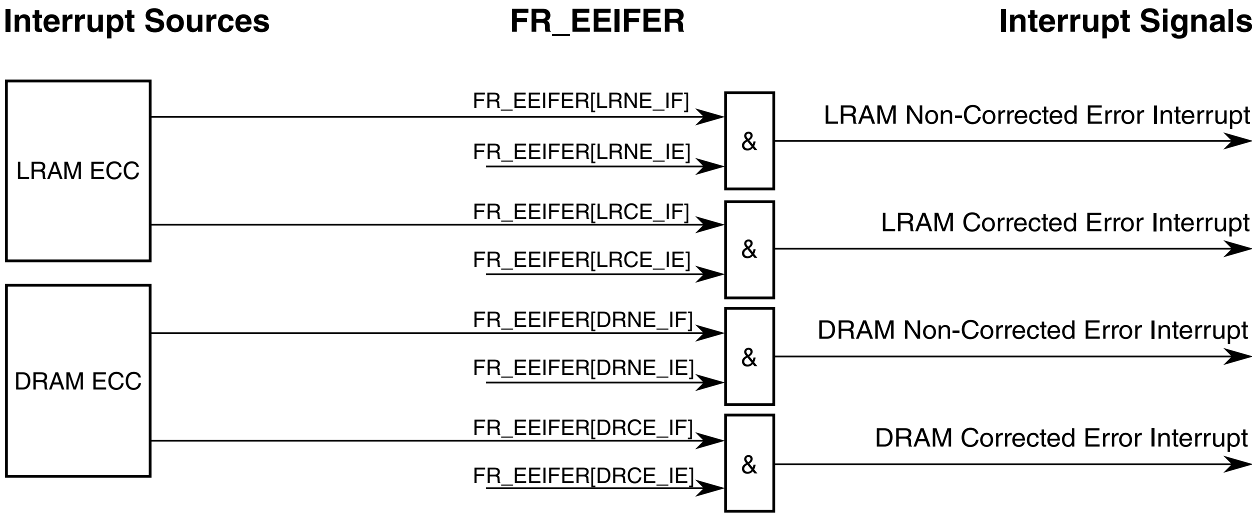


Figure 50-34. Scheme of FR_EEIFER interrupt signal generation

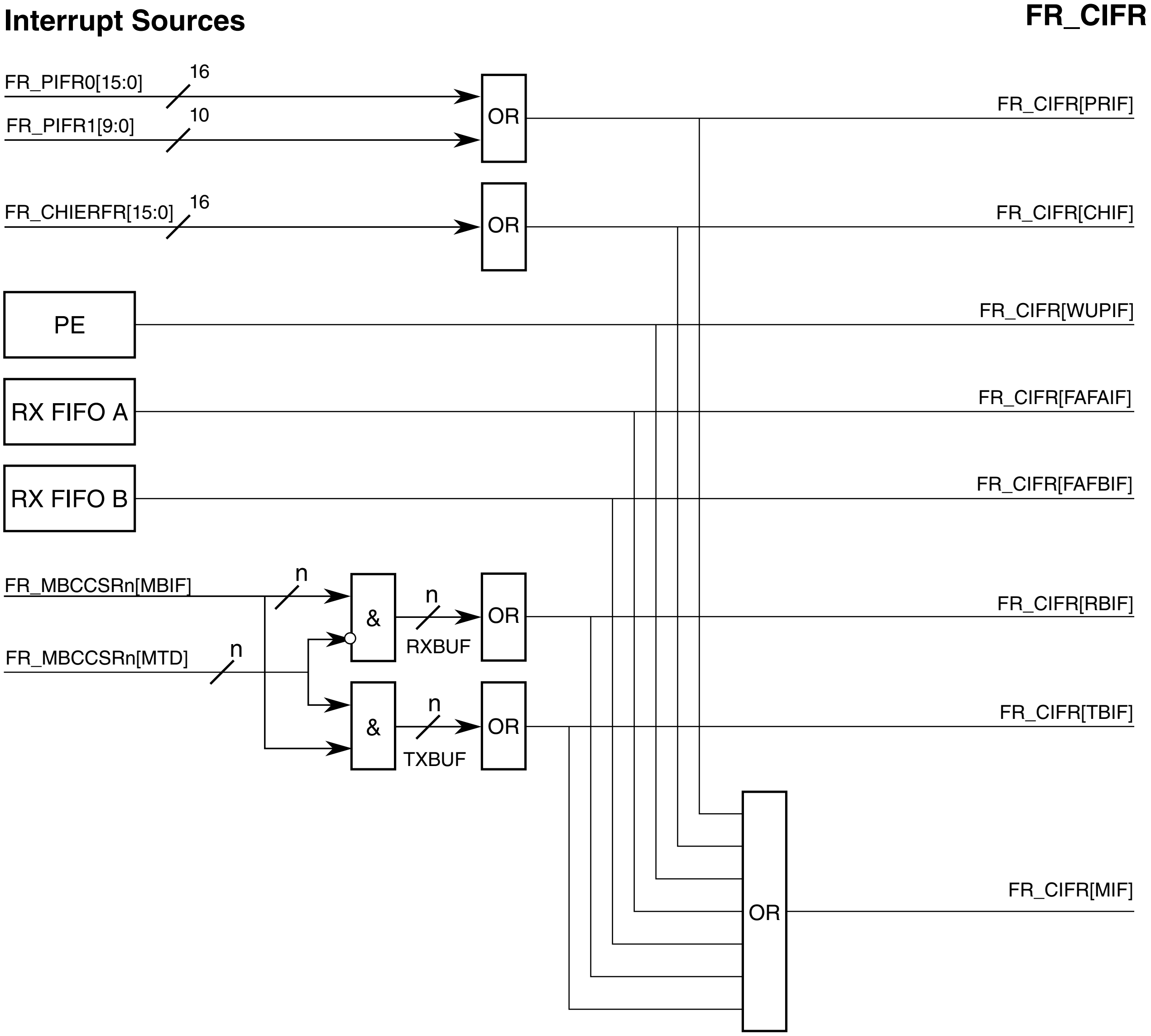


Figure 50-35. Scheme of FR_CIFR flags generation

50.7.21 Lower Bit Rate Support

The CC supports a number of lower bit rates on the FlexRay bus channels. The lower bit rates are implemented by modifying the duration of the microtick *pdMicrotick*, the number of samples per microtick *pSamplesPerMicrotick*, the number of samples per bit *cSamplesPerBit*, and the strobe offset *cStrobeOffset*. The application configures the FlexRay channel bit rate by setting the BITRATE field in the Module Configuration Register (FR_MCR). The protocol values are set internally. The available bit rates, the related BITRATE field configuration settings and related protocol parameter values are shown in the "FlexRay Channel Bit Rate Control" table below.

Table 50-55. FlexRay Channel Bit Rate Control

FlexRay Channel Bit Rate [Mbit/s]	FR_MCR[BITRATE]	pdMicrotick [ns]	gdSampleClockPeriod [ns]	pSamplesPerMicrotick	cSamplesPerBit	cStrobeOffset
10.0	000	25.0	12.5	2	8	5
8.0	011	25.0	12.5	2	10	6
5.0	001	25.0	25.0	1	8	5
2.5	010	50.0	50.0	1	8	5

Note

The bit rate of 8 Mbit/s is not defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

50.7.22 PE Data Memory (PE DRAM)

The PE Data Memory (PE DRAM) is 128 word, 16-bit wide memory with byte access, which contains the program data of the PE internal CPU. The PE DRAM is divided into two banks, 8-bit each. The memory data [7:0] are assigned to BANK0, the memory data [15:8] are assigned to BANK1.

Table 50-56. PE DRAM Layout

ADDR	BANK1	BANK0
0x00	byte1	byte0
0x01	byte3	byte2
…		
0x7F	byte255	byte254

The FlexRay module provides means to access the PE DRAM from the application. The PE DRAM application access is initiated and controlled via PE DRAM Access Register (FR_PEDRAR) and PE DRAM Access Register (FR_PEDRAR). This functionality is used to check the memory error detection.

50.7.22.1 PE DRAM Read Access

A read access from the PE DRAM can be initiated in any protocol state. The following sequence describes a read access from the PE DRAM address 0x70.

1. FR_PEDRAR:= 0x50E0;

   // INST=0x5; ADDR=070
2. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access
3. val = FR_PEDRDR[DATA];

   // read PE DRAM data

The read access is handled by the PE internal CPU with the lowest execution priority. This may cause an response delay with a maximum of 1000 PE clock cycle (25us).

50.7.22.2 PE DRAM Write Access

A write access into the PE DRAM can be initiated in any protocol state. The following sequence describes a write access to the PE DRAM address 0x70.

1. FR_PEDRDR:= DATA;

   // write value to be written into data register
2. FR_PEDRAR:= 0x30E0;

   // INST=0x3; ADDR=0x70
3. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access
4. val = FR_PEDRDR[DATA];

   // read back PE DRAM data

The write access is handled by the PE internal CPU with the lowest execution priority. This may causes an response delay with a maximum of 1000 PE clock cycle (25us).

If the conditions given in PE DRAM Write Access Limitations are fulfilled, the data provided in PE DRAM Access Regsiter (FR_PEDRAR) are written into the PE DRAM, read back in the next clock cycle and stored into the PE DRAM Access Register (FR_PEDRAR). Otherwise, data are not written into the PE DRAM and 0x0000 is stored into the PE DRAM Access Register (FR_PEDRAR).

50.7.22.3 PE DRAM Write Access Limitations

The PE DRAM is used by the protocol engine if the module is not in *POC:default config* state. The only address not used by the protocol engine is 0x70. To prevent the corruption of protocol engine data the following PE DRAM write access limitations apply for application writes.

1. When the module is in *POC:default config* state, all PE DRAM addresses are writable.
2. When the module is not *in POC:default config* state, only PE DRAM address 0x70 is writable.

50.7.23 CHI Lookup-Table Memory (CHI LRAM)

The CHI Lookup-Table Memory (CHI LRAM) is an CHI internal memory which contains the message buffer configuration data and the data field offsets for the physical message buffers. The configuration data for two message buffers or 6 data field offsets are contained in one memory row. The CHI LRAM is divided into 6 memory BANKs.

Table 50-57. CHI LRAM Layout

ADR	BANK5	BANK4	BANK3	BANK2	BANK1	BANK0
0x00	FR_MBIDXR1	FR_MBFIDR1	FR_MBCCFR1	FR_MBIDXR0	FR_MBFIDR0	FR_MBCCFR0
0x01	FR_MBIDXR3	FR_MBFIDR3	FR_MBCCFR3	FR_MBIDXR2	FR_MBFIDR2	FR_MBCCFR2
...						
0x1F	FR_MBIDXR63	FR_MBFIDR63	MBCCF63	FR_MBIDXR62	MBFID62	FR_MBCCFR62
0x20	FR_MBDOR5	FR_MBDOR4	FR_MBDOR3	FR_MBDOR2	FR_MBDOR1	FR_MBDOR0
...						
0x2B	-	-	-	-	FR_MBDOR67	FR_MBDOR66
0x2C	FR_LEETR5	FR_LEETR4	FR_LEETR3	FR_LEETR2	FR_LEETR1	FR_LEETR0

50.7.23.1 CHI LRAM Read and Write Access

The CHI LRAM is accessed by the application via regular register read and write accesses.

50.7.24 Memory Content Fault Detection

The FlexRay module provides integrated memory content error detection for both the CHI LRAM and PE DRAM, and memory content error correction for the PE DRAM. The memory error detection for the CHI LRAM uses an standard Hamming code with a Hamming distance of 3 and detects all single-bit and double-bit errors (SEDDED) . The memory error detection and correction for the PE DRAM uses an enhanced Hamming code with a Hamming distance of 4 and detects and corrects all single-bit errors and detects all double-bit errors (SECDED).

This section describes the reporting of the occurrence of memory content errors, the reaction of the module on the occurrence, and how the application can inject memory errors in order to trigger the report and response behavior.

50.7.24.1 Memory Error Types

A memory error is the distortion of one or more bits read out of the memory. The reading of the values of all zeros and all ones is considered as an special case. The FlexRay module detects and indicates the memory errors as shown in the "Detected Memory Error Types" table below. The entries on the top have higher priority.

Each memory read access reads out *all* banks of the addressed row, and runs error detection on *all* banks, even in the case that the application has triggered a read from only one bank. This may lead to the reporting of an memory error if at least one bank contains a memory error, even if an error free bank has been read.

Table 50-58. Detected Memory Error Types

Memory	Priority	Memory Data	Indication
CHI LRAM	0 (highest)	All Zero's	Non-Corrected Error
PE DRAM			Non-Corrected Error
CHI LRAM		All One's	Non-Corrected Error
PE DRAM			
CHI LRAM	1 (lowest)	One Bit Flipped	Non-Corrected Error
PE DRAM			Corrected Error

Table continues on the next page...

Table 50-58. Detected Memory Error Types (continued)

Memory	Priority	Memory Data	Indication
CHI LRAM		Two Bits Flipped	Non-Corrected Error
PE DRAM			
CHI LRAM		Three or more Bits Flipped	one out of {No error, Non-Corrected Error}, defined by coding given in CHI LRAM Checkbits and CHI LRAM Syndrome
PE DRAM			one out of {No error, Corrected Error, Non-Corrected Error}, defined by coding given in PE DRAM Checkbits and PE DRAM Syndrome

50.7.24.2 Memory Error Reporting

The memory error reporting is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

For each of the two memories exists two sets of internal registers to store the detection of one corrected and one non-corrected memory error.

If a memory error is detected, the module checks whether the related error interrupt flag in the ECC Error Interrupt Flag and Enable Register (FR_EEIFER) is set.

- *If the error interrupt flag is set, the related internal error reporting register is not updated and the related error overflow flag is set to 1 to indicate a loss of error condition.*
- *If the error interrupt flag is not set, the internal reporting register is updated and the error interrupt flag is set to 1. If two or more memory errors of the same type are detected, the error for the bank with the lower bank number will be reported, and the error overflow flag will be set to 1.*

If a memory error is detected for at least two banks of one memory, the related error overflow flag is set to 1 to indicate a loss of error condition.

In addition to above the Error indications (corrected / non corrected) along with the failing address is sent out from the module for external error logging and corrective actions by the Software.

50.7.24.2.1 PE DRAM Checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in Table 50-58. This table shows the implemented enhanced Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 50-59. PE DRAM checkbits coding

CODE	CODE				DATA							
	3	2	1	0	7	6	5	4	3	2	1	0
4[1]	X	X	X	X	X	X	X	X	X	X	X	X
3[2]	-	-	-	-	X	X	X	X	-	-	-	-
2	-	-	-	-	X	-	-	-	X	X	X	-
1	-	-	-	-	-	X	X	-	X	X	-	X
0	-	-	-	-	-	X	-	X	X	-	X	X

[1] The checkbit CODE[4] is set to 1 if and only if there is a even number of 1's in columns with X.

[2] The checkbits CODE[3]… CODE[0] are set to 1 if and only if there is a odd number of 1's in all columns with X.

This coding of the checkbit ensures that neither 0x000 nor 0xFFF are valid code words and written into the memory.

50.7.24.2.2 PE DRAM Syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in the following table.

Table 50-60. FR_EERCR[CODE] PE DRAM Syndrome Coding

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x1	0x0	No Error (Never appears in error report registers)
0x0	0x0	If data == 0: Non Corrected Error (Dedicated Handling of All Zero Code Word) If data!= 0: Corrected Error (Parity Bit 4)
0x0	0x1	Corrected Error (Parity Bit 0)
0x0	0x2	Corrected Error (Parity Bit 1)
0x0	0x3	Corrected Error (Data Bit 0)
0x0	0x4	Corrected Error (Parity Bit 2)
0x0	0x5	Corrected Error (Data Bit 1)
0x0	0x6	Corrected Error (Data Bit 2)
0x0	0x7	Corrected Error (Data Bit 3)
0x0	0x8	Corrected Error (Parity Bit 3)
0x0	0x9	Corrected Error (Data Bit 4)

Table continues on the next page...

Table 50-60. FR_EERCR[CODE] PE DRAM Syndrome Coding (continued)

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x0	0xA	Corrected Error (Data Bit 5)
0x0	0xB	Corrected Error (Data Bit 6)
0x0	0xC	Corrected Error (Data Bit 7)
0x0	0xD-0xF	Non-Corrected Error
0x1	0x1-0xF	Non-Corrected Error

50.7.24.2.3 CHI LRAM Checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in the table below. This table shows the implemented Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 50-61. CHI LRAM checkbits coding

CODE[1]	DATA															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	X	X	X	X	X	X	X	-	-	-	-
2	X	X	-	-	-	X	X	X	X	-	-	-	X	X	X	-
1	-	-	X	X	-	X	X	-	-	X	X	-	X	X	-	X
0	X	-	X	-	X	X	-	X	-	X	-	X	X	-	X	X

[1] The checkbit CODE[n] is set to 1 if and only if there is a odd number of 1's in all columns with X.

50.7.24.2.4 CHI LRAM Syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in the next table.

Table 50-62. FR_EERCR[CODE] CHI LRAM Syndrome Coding

FR_EERCR[CODE]	Description
0x00	No Error (Never appears in error report registers)
0x01-0x1F	Non Corrected Error

50.7.24.3 Memory Error Response

The memory error response is enabled only when the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

In case of the detection of a *corrected* memory error, the FlexRay module continues its normal operation using the corrected data word. This section describes the behavior of the FlexRay module after the detection of a *non-corrected* memory error.

50.7.24.3.1 CHI LRAM Error Response after CC Read

When the CC is out of the *POC:default config* state, it reads the configuration data and the data field offsets of all utilized message buffers in every slot and in the NIT. If a non-corrected memory error is detected during this module read access the error response of the module depends from LRAM location where the error occurred.

- *If the LRAM address belongs to physical message buffer configuration data the FlexRay module will consider the affected message buffer as disabled for the current search and will exclude this buffer from the search. The configuration of the affected message buffer is not changed.*

  If the affected message buffer is a tx message buffer, no frame will be transmitted from this message buffer in the next slot. If the affected message buffer is a rx message buffer, no frame will be received to this message buffer in the next slot.
- *If the LRAM address belongs to the data field offset area and the related physical message buffer is used for Rx or Tx the first access to the system memory caused by payload read or write yields to the assertion of the FR_CHIERFR[ILSA_EF]. No memory access occurs w.r.t. payload access is performed for the complete frame.*

50.7.24.3.2 CHI LRAM Error Response after Application Read

The application can read the content of the CHI LRAM via reading the FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn, FR_MBDORn, and FR_LEETRn registers. If a non-corrected memory error is detected during this kind of read access, the module indicates the detected memory error, delivers the non-corrected data read and continues its normal operation.

50.7.24.3.3 PE DRAM Error Response after CC Read

If the CC detects an non-corrected memory error during internal read of program data which is contained in PE DRAM, this is considered as an fatal protocol error and the module enters the protocol freeze state immediately.

50.7.24.3.4 PE DRAM Error Response after Application Read in *POC:default config* state

If the CC detects an non-corrected memory error during an application triggered read from any PE DRAM address and the protocol is in the *POC:default config* state, this is considered as an fatal protocol error and the module enters the protocol freeze state. This behavior allows for checking the freeze functionality in case of the detection of non-corrected errors.

50.7.24.3.5 PE DRAM Error Response after Application Read out of *POC:default config*

If the CC detects an non-corrected memory error during an application triggered read from any PE DRAM address, and the protocol is not in the *POC:default config* state, this error is not considered as an fatal error and the protocol state is not changed. This prevents any interference of the running protocol by PE DRAM error injection reads.

50.7.25 Memory Fault Injection

The error injection functionality is used by the application to inject data errors into the memories to trigger and check the memory error detection functionality.

The error injection is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) and the error injection enable control bit EIE in the ECC Error Report and Injection Control Register (FR_EERICR) are set.

The error injection mode is configured by the EIM configuration bit in the ECC Error Report and Injection Control Register (FR_EERICR).

When the error injection is enabled, each write access to the configured memory location will be distorted.

The injector has the same behavior for FlexRay module memory writes and application memory writes.

50.7.25.1 CHI LRAM Error Injection

The following sequence describes an memory error injection sequence for the CHI LRAM memory. This sequence consists of the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content. The content of the CHI LRAM is described in Table 50-57.

When the CC is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see Memory Fault Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;

   // enable ECC functionality
2. FR_EERICR[EIE]:=I_MODE;

   // configure error injection mode
3. FR_EEIAR[MID]:= 1;

   // select CHI LRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;

   // select bank for error injection; I_BANK = {0,1,2,3,4,5}
5. FR_EEIAR[ADDR]:= I_ADDR;

   // select address for error injection; I_ADDR <= 0x 2C
6. FR_EEIDR[DATA]:= D_DIST;

   // define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;

   // define checkbit distortion pattern
8. FR_EERICR[EIE]:=1;

   // enable error injection

Application Write Access:

If (I_BANK==0) -> FR_MBCCFR(2n) / FR_MBDOR(6k) / FR_LEETR0 := DATA;

If (I_BANK==1) -> FR_MBFIDR(2n) / FR_MBDOR(6k+1) / FR_LEETR1 := DATA;

If (I_BANK==2) -> FR_MBIDXR(2n) / FR_MBDOR(6k+2) / FR_LEETR2 := DATA;

If (I_BANK==3) -> FR_MBCCFR(2n+1) / FR_MBDOR(6k+3) / FR_LEETR3 := DATA;

If (I_BANK==4) -> FR_MBFIDR(2n+1) / FR_MBDOR(6k+4) / FR_LEETR4 := DATA;

If (I_BANK==5) -> FR_MBIDXR(2n+1) / FR_MBDOR(6k+5) / FR_LEETR5 := DATA;

// write DATA to the defined injection bank and injection address (see Table 50-57).

50.7.25.2 PE DRAM Error Injection

The following sequence describes an memory error injection sequence for the PE DRAM memory. This sequence consists of the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content.

When the FlexRay module is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see PE DRAM Error Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;

   // enable ECC functionality
2. FR_EERICR[EIE]:=I_MODE;

   // configure error injection mode
3. FR_EEIAR[MID]:= 0;

   // select PE DRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;

   // define bank for error injection; I_BANK = {0,1}
5. FR_EEIAR[ADDR]:= I_ADDR;

   // define address for error injection; I_ADDR <= 0x7F
6. FR_EEIDR[DATA]:= D_DIST;

   // define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;

   // define checkbit distortion pattern
8. FR_EERICR[EIE]:=1;

   // enable error injection

Application Write Access (e.g. I_ADDR=0x70):

1. FR_PEDRAR:= 0x30E0;

   // INST=0x3; ADDR=0x70

2. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

3. val = FR_PEDRDR[DATA]; |

   // get read back PE DRAM data

Note

> The write access to the PE DRAM triggers an subsequent read access from PE DRAM in the next cycle, which triggers the detection of the distorted data.

50.8 Application Information

Module operation is discussed in this section.

50.8.1 Module Configuration

This section describes essential parts of the module configuration.

50.8.1.1 Configure System Memory Access Time-Out Register (FR_SYMATOR)

To ensure reliable operation of the CC, the application must ensure that the TIMEOUT value in System Memory Access Time-Out Register (FR_SYMATOR) and the CHI clock frequency f_{CHI} in MHz fulfill this equation[1] :

$$0 \leq \mathrm{SYMATOR}\left[\mathrm{TIMEOUT}\right] \leq \left\lfloor 0.45 \bullet f_{\mathrm{CHI}} - 8 \right\rfloor$$

Equation 48

[1] See Controller Host Interface Clocking for all constraints of minimum CHI clock frequency.

For a given SYMATOR[TIMEOUT] value, f_{CHI} can be increased without causing unreliable operation of the CC. The same holds for reducing the SYMATOR[TIMEOUT] value for a given f_{CHI}.

Some examples for maximum values of the SYMATOR[TIMEOUT] for a minimum CHI frequency are given in the table below.

Table 50-63. Maximum SYMATOR[TIMEOUT] examples

f_{CHI}	SYMATOR[TIMEOUT]	f_{CHI}	SYMATOR[TIMEOUT]
>= 18 MHz	0	>= 100 MHz	<= 37
>= 23 MHz	<= 2	>= 120 MHz	<= 46
>= 27 MHz	<= 4	>= 140 MHz	<= 55
>= 32 MHz	<= 6	>= 160 MHz	<= 64
>= 60 MHz	<= 19	>= 180 MHz	<= 73
>= 80 MHz	<= 28	>= 200 MHz	<= 82

50.8.1.1.1 System Bus Wait State Constraints

The SYMATOR[TIMEOUT] value corresponds directly to a certain acceptable number of wait states on the system bus.

For single channel configurations and if the sync frame table generation functionality is *not* used (FR_SFTCCSR[SDVEN,SIDEN] = 0) no timeout will be detected if less than 2 × SYMATOR[TIMEOUT]+1 wait states will be seen on the system bus for each system bus access.

For dual channel configurations, or if the sync frame table generation functionality is used, no timeout will be detected if less than SYMATOR[TIMEOUT]-1 wait states will be seen on the system bus for each system bus access.

50.8.1.2 Configure Data Field Offsets

The data field offsets are located in the Message Buffer Data Field Offset Registers (FR_MBDORn) and Receive FIFO Start Data Offset Register (FR_RFSDOR). The application has to configure the data field offset values for all message buffers which are used.

When the module is enabled, the initilization value of the FR_MBDORn[MBDO] and FR_RFSDOR[SDO] is 0. This value is considered to be illegal (see System Bus Illegal Address Access).

50.8.2 Initialization Sequence

This section describes the required steps to initialize the CC. The first subsection describes the steps required after a module reset, the second section describes the steps required after preceding shutdown of the CC.

50.8.2.1 Module Initialization

This section describes the module related initialization steps after a system reset.

1. Configure CC.
   a. configure the control bits in the Module Configuration Register (FR_MCR)
   b. configure the system memory base address in System Memory Base Address Register (FR_SYMBADR)
2. Enable the CC.
   a. write 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR)

The CC now enters the Normal Mode. The application can commence with the protocol initialization described in Protocol Initialization.

50.8.2.2 Protocol Initialization

This section describes the protocol related initialization steps.

1. Configure the Protocol Engine.
   a. issue CONFIG command via Protocol Operation Control Register (FR_POCR)
   b. wait for *POC:config* in Protocol Status Register 0 (FR_PSR0)
   c. configure the FR_PCR0,…, FR_PCR30 registers to set all protocol parameters
2. Configure the Message Buffers and FIFOs.
   a. set the number of message buffers used and the message buffer segmentation in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

b. define the message buffer data size in the Message Buffer Data Size Register (FR_MBDSR)

c. configure each message buffer by setting the configuration values in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn), Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn), Message Buffer Frame ID Registers (FR_MBFIDRn), Message Buffer Index Registers (FR_MBIDXRn)

d. configure the FIFOs

e. issue CONFIG_COMPLETE command via Protocol Operation Control Register (FR_POCR)

f. wait for *POC:ready* in Protocol Status Register 0 (FR_PSR0)

After this sequence, the CC is configured as a FlexRay node and is ready to integrate into the FlexRay cluster.

50.8.2.3 CHI LRAM Initialization

The initialization of the CHI LRAM is performed by the CC when it leaves the Disabled Mode. The unitization runs for 45 CHI clock cycles. All fields in the FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBDORn, and LEETRn registers are initialized to 0. All application read or write accesses to these registers are delayed until the initialization is finished.

50.8.2.4 PE DRAM Initialization

The PE DRAM initialization is performed by the CC in the *POC:default config* state. This initialization runs for 4.8 μs, and will delay the state transition from *POC:default config* into *POC:config*.

50.8.3 Memory Fault Injection out of POC:default config

This section provides information for application driven memory fault injection out if *POC:default config*. The CC provides means to inject memory faults from the application without any impacts to the internal protocol operation of the CC.

50.8.3.1 CHI LRAM Error Injection out of POC:default config

The CC will never perform any internal read access from the LRAM ECC Error Test Registers (FR_LEETRn). Any memory errors injected into these CHI LRAM locations will never be detected by internal access, independent from the protocol state.

The application should use these registers and related CHI LRAM location to inject memory errors into the CHI LRAM. The injection sequence is described in CHI LRAM Error Injection.

50.8.3.2 PE DRAM Error Injection out of POC:default config

The CC will never perform any internal read access from the PE DRAM address 0x70. This is the only one PE DRAM address writable by the application out of the *POC:default config* state.

The application should use these PE DRAM location to inject memory errors into the PE DRAM. The injection sequence is described in PE DRAM Error Injection.

50.8.4 Shut Down Sequence

This section describes a secure shut down sequence to stop the CC gracefully. The main targets of this sequence are

- *finish all ongoing reception and transmission*
- *do not corrupt FlexRay bus and do not disturb ongoing FlexRay bus communication*

For a graceful shutdown the application shall perform the following tasks:

1. Disable all enabled message buffers.
    a. repeatedly write 1 to FR_MBCCSRn[EDT] until FR_MBCCSRn[EDS] == 0.
2. Stop Protocol Engine.
    a. issue HALT command via Protocol Operation Control Register (FR_POCR)
    b. wait for *POC:halt* in Protocol Status Register 0 (FR_PSR0)

50.8.5 Number of Usable Message Buffers

This section describes the required minimum CHI clock frequency for a specified number of utilized message buffers configured in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR), a configured mini-slot length *gdMinislot*, and a configured nominal macrotick length *gdMacrotick*[1].

Additional constraints for the minimum CHI clock frequency are given in Controller Host Interface Clocking.

The CC uses a sequential search algorithm to determine the individual message buffer assigned or subscribed to the next slot. This search is started at the start of slot and must be finished before the start of the next slot.

The shortest FlexRay slot is a corrected empty dynamic slot. An corrected empty dynamic slot is a mini-slot and consists of gdMinislot corrected macroticks with a duration of *gdMacrotick*. The minimum duration of an corrected macrotick is $gdMacrotick_{\text{min}}$ = 39 μT. This results in a minimum length of an correct slot

$$\Delta_{\text{slotmin}} = 39 \bullet \text{pdMicrotick} \bullet \text{gdMinislot}$$

Equation 49

The message buffer search engine runs on the CHI clock and evaluates one individual message buffer per CHI clock cycle. For internal status update operations and to account for clock domain crossing jitter, an additional amount of 27 CHI clock cycles is required to ensure correct search engine operation.

For a given number of utilized message buffers FR_MBSSUTR[LAST_MB_UTIL] + 1 and for a given CHI clock frequency f_{chi}, this results in a search duration of

$$\Delta_{\text{search}} = \frac{1}{f_{\text{chi}}} \bullet (\text{FR\_MBSSUTR}[\text{LAST\_MB\_UTIL}] + 27)$$

Equation 50

The message buffer search must be finished within one slot which requires fulfillment of this equation::

$$\Delta_{\text{search}} \leq \Delta_{\text{slotmin}}$$

Equation 51

This results in the formula given below which determines the required minimum CHI frequency for a given number of message buffers that are utilized.

[1] See Controller Host Interface Clocking for all constraints of minimum CHI clock frequency.

$$f_{\text{chi}} \geq \frac{(\text{FR\_MBSSUTR}[\text{LAST\_MB\_UTIL}]+27)}{39 \bullet \text{pdMicrotick} \bullet \text{gdMinislot}}$$

Equation 52

The required minimum CHI Clock frequency for a selected set of relevant protocol parameters and for the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR) set to 63 is given in the following table.

Table 50-64. Minimum fchi [MHz] examples (64 message buffers used)

pdMicrotick [ns]	***gdMinislot***					
	2	**3**	**4**	**5**	**6**	**7**
25.0	46.7	31.2	23.4	18.7	15.6	13.4
50.0	23.4	15.6	11.7	9.4	7.8	6.7

Note

If the minimum CHI frequency is not met the CHIERFR[MBS_EF] flag is set. Refer to the CHI Error Flag Register (FR_CHIERFR) for details.

50.8.6 Protocol Control Command Execution

This section considers the issues of the protocol control command execution.

The application issues any of the protocol control commands listed in the POCCMD field of the FR_POCR Field Descriptions table, located in the Register Descriptions section, by writing the command to the POCCMD field of the Protocol Operation Control Register (FR_POCR). As a result the CC sets the BSY bit while the command is transferred to the PE. When the PE has accepted the command, the BSY flag is cleared. All commands are accepted by the PE.

The PE maintains a protocol command vector. For each command that was accepted by the PE, the PE sets the corresponding command bit in the protocol command vector. If a command is issued while the corresponding command bit is set, the command is not queued and is lost.

If the command execution block of the PE is idle, it selects the next accepted protocol command with the highest priority from the current protocol command vector according to the protocol control command priorities given in the next table. If the current protocol state does not allow the execution of this protocol command (see POC state changes in *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*) the CC asserts the illegal protocol command interrupt flag IPC_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). The protocol command is not executed in this case.

Some protocol commands may be interrupted by other commands or the detection of a fatal protocol error as indicated by the table below. If the application issues the FREEZE or READY command, or if the PE detects a fatal protocol error, some commands already stored in the command vector will be removed from this vector.

Table 50-65. Protocol Control Command Priorities

Protocol Command	Priority	Interrupted By	Cleared and Terminated By
FREEZE	(highest) 1	none	
READY	2		
CONFIG_COMPLETE	3		
ALL_SLOTS	4	FREEZE, READY, CONFIG_COMPLET, fatal protocol error	FREEZE, READY, CONFIG_COMPLETE,fatal protocol error
ALLOW_COLDSTART	5		
RUN	6		FREEZE, fatal protocol error
WAKEUP	7		FREEZE, fatal protocol error
DEFAULT_CONFIG	8		FREEZE, fatal protocol error
CONFIG	9		
HALT	(lowest) 10		FREEZE, READY, CONFIG_COMPLETE,fatal protocol error

50.8.7 Message Buffer Search On Simple Message Buffer Configuration

This sections describes the message buffer search behavior for a simplified message buffer configuration. The FIFO behavior is not considered in this section.

50.8.7.1 Simple Message Buffer Configuration

A simple message buffer configuration is a configuration that has at most one transmit message buffer and at most one receive message buffer assigned to a slot *S*. The simple configuration used in this section utilizes two message buffers, one single buffered transmit message buffer and one receive message buffer.

The transmit message buffer has the message buffer number *t* and has following configuration

Table 50-66. Transmit Buffer Configuration

Register	Field	Value	Description
FR_MBCCSR*t*	MTD	1	transmit buffer

Table continues on the next page...

Table 50-66. Transmit Buffer Configuration (continued)

Register	Field	Value	Description
FR_MBCCFR*t*	MTM	0	event transition mode
	CHA	1	assigned to channel A
	CHB	0	not assigned to channel B
	CCFE	1	cycle counter filter enabled
	CCFMSK	000011	cycle set = {4n} = {0,4,8,12,…}
	CCFVAL	000000	
FR_MBFIDR*t*	FID	*S*	assigned to slot S

The availability of data in the transmit buffer is indicated by the commit bit FR_MBCCSRt[CMT] and the lock bit FR_MBCCSRt[LCKS].

The receive message buffer has the message buffer number r and has following configuration

Table 50-67. Receive Buffer Configuration

Registe*r*	Field	Value	Description
FR_MBCCSR*r*	MTD	0	receive buffer
FR_MBCCFR*r*	MTM	-	n/a
	CHA	1	assigned to channel A
	CHB	0	not assigned to channel B
	CCFE	1	cycle counter filter enabled
	CCFMSK	000001	cycle set = {2n} = {0,2,4,6,…}
	CCFVAL	000000	
FR_MBFIDR*r*	FID	*S*	subscribed slot

Furthermore the assumption is that both message buffers are enabled (FR_MBCCSRt[EDS] = 1 and FR_MBCCSRr[EDS] = 1)

Note

The cycle set {4n+2} = {2,6,10,…} is assigned to the receive buffer only.

The cycle set {4n} = {0,4,8,12,…} is assigned to both buffers.

50.8.7.2 Behavior in static segment

In this case, both message buffers are assigned to a slot *S* in the *static* segment.

The configuration of a transmit buffer for a static slot *S* assigns this slot to the node as a transmit slot. The FlexRay protocol requires:

- When a slot occurs, if the slot is assigned to a node on a channel that node must transmit either a normal frame or a null frame on that channel. Specifically, a null frame will be sent if there is no data ready, or if there is no match on a transmit filter (cycle counter filtering, for example).

Regardless of the availability of data and the cycle counter filter, the node will transmit a frame in the static slot *S*. In any case, the result of the message buffer search will be the transmit message buffer *t*. The receive message buffer *r* will not be found, no reception is possible.

50.8.7.3 Behavior in dynamic segment

In this case, both message buffers are assigned to a slot *S* in the *dynamic* segment. The FlexRay protocol requires:

- When a slot occurs, if a slot is assigned to a node on a channel that node only transmits a frame on that channel if there is data ready and there is a match on relevant transmit filters (no null frames are sent).

The transmission of a frame in the dynamic segment is determined by the availability of data and the match of the cycle counter filter of the transmit message buffer.

50.8.7.3.1 Transmit Data Not Available

If transmit data are *not available*, i.e. the transmit buffer is not committed FR_MBCCSR*t*[CMT]=0 and/or locked FR_MBCCSR*t*[LCKS]=1,

1. for the cycles in the set {4n}, which is assigned to both buffers, the receive buffer will be found and the node can receive data, and
2. for the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive cycles are shown in the following figure.

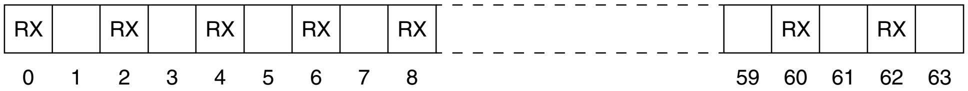


Figure 50-36. Transmit Data Not Available

50.8.7.3.2 Transmit Data Available

If transmit data are *available*, i.e. the transmit buffer is committed FR_MBCCSR*t*[CMT]=1 and not locked FR_MBCCSR*t*[LCKS]=0,

1. for the cycles in the set {4n}, which is assigned to both buffers, the transmit buffer will be found and the node transmits data.
2. for the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive and transmit cycles are shown in the following figure.

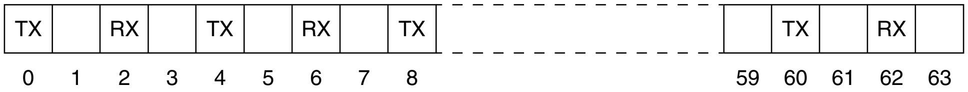


Figure 50-37. Transmit Data Available

Chapter 51
SENT Receiver (SRX)

51.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Single Edge Nibble Transmission (SENT) Receiver (SRX) module is a multi-channel receiver for receiving serial data frames which are being transmitted by a sensor implementing the SENT encoding scheme and present them to the CPU for further processing.

This module is based on the SAE J2716 information report titled "SENT - Single Edge Nibble Transmission for Automotive Applications" and released on January 27, 2010 (http://www.sae.org), Apr2007 and Feb2008. As per this standard, the SENT protocol is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/Ds and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

The SENT encoding scheme is a unidirectional communications scheme from the sensor / transmitting device to the controller /receiving device which does not include a coordination signal from the controller/receiving device. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

51.1.1 Features

The SENT receiver module has the following features:

- The number of SENT channels supported are device-specific. See the device configuration details.
- Unified message read logic to transfer all messages received to system memory via interrupts or DMA
  - Programmable option per channel to support reading of messages via DMA or Interrupt
  - Internal round robin arbitration sequencer to loop across channels to read messages from each channel leaving CPU/DMA transparent to this process
  - A FIFO for storing the received Fast Messages from all channels that are to be read out via DMA. FIFO depth is device-specific. See the device configuration section for details.
  - All received messages are stored excluding the synchronization pulse (for Fast messages) and identifiers (for Slow Serial Messages)
  - Separate buffers for storing Fast and Slow Serial Messages
  - Separate DMA and Interrupts generated for Fast and Slow Serial Messages
- Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel
  - Calibration pulse correction factor per channel is available for use by software
- Implements for each channel all receiver diagnostics as specified by the SAE Specifications
- Internal 4-bit CRC calculator that can be configured to compute CRC in both Legacy and Recommended method (as given in SAE SENT Specification) for both Fast and Short Serial Message per channel
- Internal 6-bit CRC calculator for Enhanced Serial Messages per channel
- Supports all bit rates by having per channel configurable Rx clock periods from 3 µs to 90 µs
- Support for configurable number of data nibbles per channel
- Option to disable individual channel via programmable control bit
- Time stamping on start of each valid message (i.e. without errors) across all channels. This time stamp is appended to each message for synchronization of messages and for application to understand how much time has elapsed between successive sensor readings

- Support for pause pulse per channel. This can be enabled by programming a bit in the control register for that channel
- Input Programmable Filter on each channel to filter any glitches on input. This filter duration is programmable by the user software.
- 32-bit register interface for programming the module's registers and reading the received messages
- Works on two clocks, one for accurate receive operations (high frequency receiver clock or protocol clock) and other for message read logic (system bus clock) and the module can work irrespective of the frequency relationship between the two clocks. See Clocks and resets for more details on the clocks.

51.1.2 Modes of operation

Modes of operation.

51.1.2.1 Debug mode

In debug mode, message reception will be halted (without waiting for the current message being received to be completed) and all existing status and messages will be retained. DBG_FRZ bit is required to be set for this mode. On debug exit, a calibration length error (CAL_LEN_ERR) can be reported and the channels will start receiving messages from a valid calibration pulse.

51.1.2.2 Low power modes

SENT Receiver behavior is affected by chip-specific low power modes (for example, STOP or WAIT modes). These low power modes should either gate-off both the system bus clock and the protocol clocks or reduce their frequencies, depending on the application.

On exiting from these low power modes, calibration length error (CAL_LEN_ERR) can get set in channel status registers as the SENT sensor (transmitter) could be in the middle of a message transmission at that point. Thus, software should clear all error status bits after exiting the low power modes.

Modes where both system bus clock and protocol clock are gated: In this case, both message reception and interrupt / DMA request generation will halt. The internal receiver channels will freeze (since clock is not there) and will not receive any messages. Messages which are completely received before entering the low power mode can be read after the low power mode exit.

Modes where frequency of system bus clock is changed: For reduced power consumption the system bus clock can be reduced but not below a minimum value (See System bus clock requirements) to avoid overflow of received messages. Protocol clock should not go below the minimum value (See High frequency receiver clock (protocol clock) requirements) for accurate receive operations.

Note

Gating of clocks in the low power mode is chip-specific, see the chapter that refers how the modules behave in low power modes.

NOTE

If the SOC requests entry to debug or stop mode, the message being received currently is discarded and the SOC enters debug/ stop mode immediately. This does not affect the messages received completely prior to entering debug mode and will still be present on the message buffer and registers until they are read out.

NOTE

The message read registers [Channel ‘n' Fast Message Data Read Register (n = 0 to (CH-1)) (CHn_FMSG_DATA), Channel ‘n' Fast Message CRC Read Register (n = 0 to (CH-1)) (CHn_FMSG_CRC), Channel ‘n’ Fast Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_FMSG_TS), Channel ‘n’ Serial Message Read Register (Bit 3) (n = 0 to (CH-1)) (CHn_SMSG_BIT3), Channel ‘n’ Serial Message Read Register (Bit 2) (n = 0 to (CH-1)) (CHn_SMSG_BIT2), Channel ‘n’ Serial Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_SMSG_TS), DMA Fast Message Data Read Register (DMA_FMSG_DATA), DMA Fast Message CRC Read Register (DMA_FMSG_CRC), DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS), DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3), DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2) and

DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)] will appear to lose their contents in the following conditions:

- The very first message is being received but not yet completely received and the MCU enters debug or freeze mode, the current message reception will get discarded and message read registers (as mentioned above) will read zeros
- Auto clear functionality is enabled (GBL_CTRL[FAST_CLR] = 1). In this case, when first message is read, it will get clear the message read registers (due to auto clear functionality being enabled). On reading again, the message read registers (as mentioned above) might read zeros.

If the MCU requests entry to debug or stop mode, the message being received currently by SENT Receiver is discarded and the MCU enters debug/stop mode immediately. This does not affect the messages received completely, prior to entering debug mode and these messages will still be present on the message buffer and registers until they are read out. Thus allow one message to be received completely and do not enable "Auto Clear" (GBL_CTRL[FAST_CLR] = 0), to allow messages to be read in debug or stop mode.

51.1.3 Block diagram

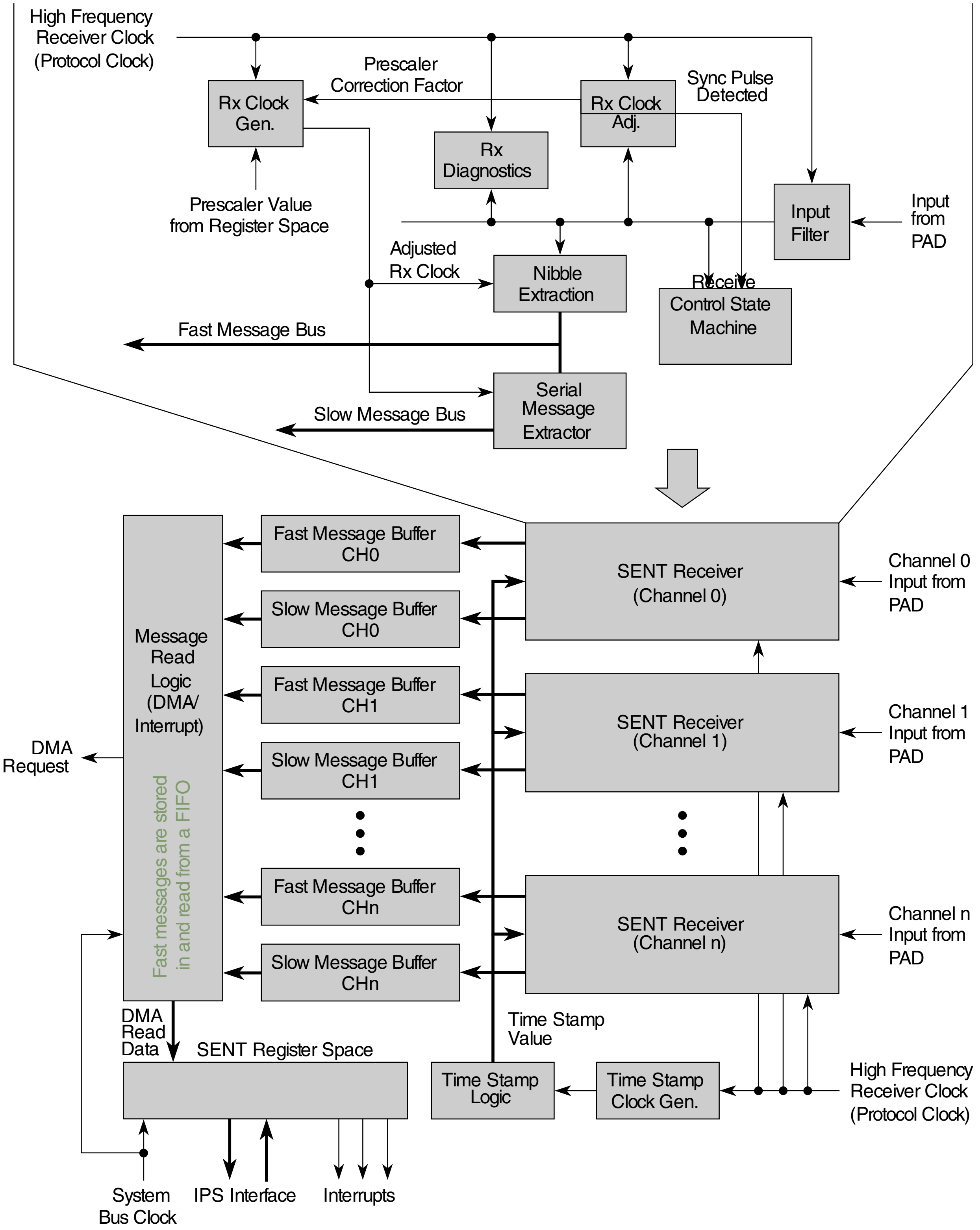


Figure 51-1. SENT Receiver block diagram

51.1.4 Design overview

The SENT Receiver Module is a unified receiver module comprising of a number of receiver channels and a unified DMA and Register Space for accessing the messages from each channel. Figure 51-1 shows the top level block diagram for the SENT Receiver module.

Each channel of the SENT Receiver is composed of a Clock Generation and Adjustment block to generate the receiver clock and adjust it for frequency variation in the transmitter clock. This block also detects the synchronization/calibration pulse which is present in every SENT message.

The SENT Receiver State Machine controls the sequencing of decoding the messages and combining the information as nibbles. In parallel a diagnostic block is running that monitors the incoming messages for any errors and also verifies the CRC of the received messages. The fast and serial message extractors collate the message nibbles and add the channel number and time stamp information to the messages.A free running time stamp counter is used to add the time stamp value to each message received. The value of this counter is input to each channel and the channel logic appends the time stamp to each message. These messages are then formatted to be stored in the DMA FIFOs (for channels which are selected to be connected to the DMA) or in corresponding message read buffer. Only the messages that pass the diagnostics checks are forwarded to be stored in the FIFOs or buffers.

The SENT Receiver provides two mechanisms for reading the received messages, which are via DMA transfers and via Interrupts. Based on the control bit settings in the control registers, a DMA request or Interrupt is asserted to signal the DMA controller or CPU that messages are available to be read.

The free running time stamp counter, register block and DMA interface are common for all channels configured. The register space contains all the necessary registers for all channels. The register space is 32-bit aligned and accessible via 8-/16-/32-bit accesses. The registers for all un-configured channels are marked as reserved.

51.1.4.1 Functional overview

After reset, the SENT module comes up in the disabled state. All channel enable bits and the global enable bit are set to 0 and the user software must program the SENT module parameters correctly before enabling it. The user software must correctly configure all parameters to avoid any underflow or overflow in the message FIFOs or buffers.

Once enabled, the Receiver Clock Generation and Adjustment block detects the synchronization pulse. Until then, the state machine blocks all subsequent operations of the receiver. Once the synchronization pulse is detected, the receiver clock tick period is compensated for variation in transmitter clock tick period and the state machine then waits for the reception of data nibbles. The adjusted receiver clock is used to sample the data nibbles from the incoming serial message. The state machine waits until the status and communication nibble and configured number of data nibbles are received. The state machine then waits for the pause pulse (if configured to receive a pause pulse). The fast and serial message extractor block provides the state machine with the information on how many nibbles have been received. All diagnostics checks are computed in parallel. Once all required data nibbles are received, the CRC is checked, time stamp and channel number is added to the message. The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer along with the message. This final message is stored in the DMA FIFO or message buffer as described in the subsequent sections. Any error detected during the receive process causes the status bits in the corresponding channel's status register to be set.

The above process repeats across each enabled channel in parallel and keeps pushing error free messages into the FIFOs or buffers on high frequency receiver clock. The module's message read logic synchronizes all these messages on the bus clock and pops out these messages on any DMA or CPU read access. Any DMA underflow errors are stored in the global status register while the overflow errors are stored in the corresponding channel's status register. In case of an overflow in Fast Messages, the newer messages received will overwrite the previous message.

For detailed description on the complete functionality of each block, see Functional description.

51.2 External signal description

This section lists all inputs to the SENT Receiver block.

51.2.1 SENT_RX [(CH-1):0]

These pins serve as the serial data input for each of the SENT Receiver channel. There is one bit per channel supported by the device. The SENT sensor input pads on the device using this module should have pull-up enabled to ensure a 1'b1 is seen at the module input when SENT sensor is not driving any data.

Note

The value of CH is device-specific. See the device configuration details.

51.3 Memory map and register definition

This section provides a detailed description of all registers accessible in the SENT Receiver block.

The following table gives an overview on all SENT Receiver registers. The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s. The Memory Map and complete module is in Big Endian Format.

The module will not check for correctness of programmed values in Register Space and software must ensure that correct values are being written.

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level. The number of registers are device specific. See the device configuration section for details.

NOTE

The value of the CH is device-specific. See the device configuration section for details.

SRX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Control Register (SRX_GBL_CTRL)	32	R/W	0001_0010h	51.3.1/1889
4	Channel Enable Register (SRX_CHNL_EN)	32	R/W	0000_0000h	51.3.2/1891
8	Global Status Register (SRX_GBL_STATUS)	32	w1c	0000_0000h	51.3.3/1892
C	Fast Message Ready Status Register (SRX_FMSG_RDY)	32	R/W	0000_0000h	51.3.4/1893
10	Slow Serial Message Ready Status Register (SRX_SMSG_RDY)	32	R/W	0000_0000h	51.3.5/1894
18	Data Control Register 1 (SRX_DATA_CTRL1)	32	R/W	1100_0000h	51.3.6/1895

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	Fast Message DMA Control Register (SRX_FDMA_CTRL)	32	R/W	0000_0000h	51.3.7/1896
2C	Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)	32	R/W	0000_0000h	51.3.8/1897
34	Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)	32	R/W	0000_0000h	51.3.9/1898
38	Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)	32	R/W	0000_0000h	51.3.10/ 1899
40	DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)	32	R	0000_0000h	51.3.11/ 1900
44	DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)	32	R	0000_0000h	51.3.12/ 1901
48	DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)	32	R	0000_0000h	51.3.13/ 1901
50	DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)	32	R	0000_0000h	51.3.14/ 1902
54	DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)	32	R	0000_0000h	51.3.15/ 1903
58	DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)	32	R	0000_0000h	51.3.16/ 1904
60	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH0_CLK_CTRL)	32	R/W	0000_8000h	51.3.17/ 1904
64	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH0_STATUS)	32	w1c	0000_0000h	51.3.18/ 1906
68	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH0_CONFIG)	32	R/W	0000_0104h	51.3.19/ 1909
70	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH1_CLK_CTRL)	32	R/W	0000_8000h	51.3.17/ 1904
74	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH1_STATUS)	32	w1c	0000_0000h	51.3.18/ 1906
78	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH1_CONFIG)	32	R/W	0000_0104h	51.3.19/ 1909
160	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_DATA)	32	R	0000_0000h	51.3.20/ 1912
164	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_CRC)	32	R	0000_0000h	51.3.21/ 1913
168	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_TS)	32	R	0000_0000h	51.3.22/ 1914
16C	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH0_SMSG_BIT3)	32	R	0000_0000h	51.3.23/ 1914
170	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH0_SMSG_BIT2)	32	R	0000_0000h	51.3.24/ 1915
174	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH0_SMSG_TS)	32	R	0000_0000h	51.3.25/ 1916
178	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_DATA)	32	R	0000_0000h	51.3.20/ 1912

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17C	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_CRC)	32	R	0000_0000h	51.3.21/ 1913
180	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_TS)	32	R	0000_0000h	51.3.22/ 1914
184	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH1_SMSG_BIT3)	32	R	0000_0000h	51.3.23/ 1914
188	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH1_SMSG_BIT2)	32	R	0000_0000h	51.3.24/ 1915
18C	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH1_SMSG_TS)	32	R	0000_0000h	51.3.25/ 1916

51.3.1 Global Control Register (SRX_GBL_CTRL)

This register provides the global control and setting for the SENT Receiver Module. The SENT_EN bit must be programmed after all other modules settings and control bits have been written to.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSPRSC								0			FIFOWM				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0				0	0	0	0	FIFO_EN	0	FAST_CLR	0	DBG_FRZ	0	SENT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

SRX_GBL_CTRL field descriptions

Field	Description
0–7 TSPRSC	Time Stamp Prescaler Value. Value in number of clock cycles of high frequency receiver clock set in order to obtain a clock frequency for the time stamp counter block. 0 to 255 - Time Stamp Clock = (High Frequency Clock)/(Value + 1). Value set by user software should be such that the clock period is at least 1 µs. If the high frequency clock input is 75 MHz, then the value set in this field would be 0x4A (or 74) to obtain a 1 µs clock.

Table continues on the next page...

SRX_GBL_CTRL field descriptions (continued)

Field	Description
	Note: Since the time stamp clock is common for all channels, the timestamp clock frequency must be at least twice the fastest receiver channel in the application. This is to ensure that the time stamp clock is fast enough to show change in timestamp from message to message.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 FIFOWM	DMA Read FIFO Water Mark Level. Indicates the number of Fast Messages to be stored in FIFO before a DMA request is asserted. Default = 1 message. 0 Invalid Value. Should not be used 1 to 31 Valid water mark levels. Value fixed by user software. **NOTE:** The number of bytes to be transferred set in the DMA Controller's TCD must be set equal to the number of bytes corresponding to the set water mark value.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FIFO_EN	FIFO enable bit. Enables FIFO when DMA is enabled. When this bit is low, data is read from module as explained inDMA request for Fast Message reading when FIFO is disabled and when the bit is high, data is read from module as explained inDMA request for Fast Message reading via FIFO . Default value is 0. 0 FIFO is disabled 1 FIFO is enabled
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FAST_CLR	Fast Clearing Enable bit. Enables clearing of ready status (for both fast and slow) bit automatically when corresponding message is read. Default value is 1. SeeFast Message Ready Status Register (SRX_FMSG_RDY) andSlow Serial Message Ready Status Register (SRX_SMSG_RDY) for details on the ready status bits. 0 Fast Clearing is disabled. Bits will be cleared by writing 1 1 Fast Clearing is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DBG_FRZ	Debug Freeze. This bit will enable the debug mode support. SENT Module will freeze in debug mode if this bit is set. Default is 0. 0 No effect in debug mode 1 Freeze in debug mode
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRX_GBL_CTRL field descriptions (continued)

Field	Description
31 SENT_EN	SENT Receiver Global Enable. This bit enables or disables the complete receiver module irrespective of the setting of individual channel enable bits. User software must program all channel parameters before setting this bit. Default is 0. 0 Entire module is disabled 1 Module is enabled

51.3.2 Channel Enable Register (SRX_CHNL_EN)

This register is used to enable individual SENT Receiver channels in the module. The user software must ensure all channel parameters and control settings have been programmed before writing to this bit. In case these parameters are changed while the channel is enabled, normal operation is not guaranteed.

NOTE

Reads of bits beyond those for the supported number of channels should be ignored, and these bits must be written to 0.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														EN_ CH	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CHNL_EN field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 EN_CH	Enable bits for channels 1 to 0 . All channels are disabled after reset. User software must program all channel parameters before setting this bit. Once set, other channel parameters in registers should not be changed until this bit is cleared. Only Interrupt Enable and Status bits can be written to. 0 Channel is disabled 1 Channel is enabled. Channel parameter registers are locked for writing

51.3.3 Global Status Register (SRX_GBL_STATUS)

This register provides the status of various conditions in the module that are common across all channels.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					FMFO	FMDU	SMDU	0							
W						w1c	w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_GBL_STATUS field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 FMFO	Fast Message FIFO Overflow: Interrupt status bit which indicates that Fast Message is received on any channel which has DMA enabled after FIFO is full. Default value is 0. This bit is valid only when fifo is enabled 0 No Overflow 1 Overflow occurred
22 FMDU	Fast Message DMA Underflow: Interrupt Status bit that indicates when there is an underflow in Fast Message buffers or FIFO in any channel which has DMA enabled. If FIFO is enabled, it is set when FIFO is empty and software still reads Fast Message DMA registers. And if FIFO is not enabled, it is set when there is no message in the DMA buffers for read and the software still reads the Fast Message DMA register. Default value is 0. 0 No underflow 1 Underflow occurred
23 SMDU	Slow Serial Message DMA Underflow: Interrupt Status bit that indicates when there is an Underflow in Slow Serial Message DMA buffers on channel which has DMA enabled. The bit is set when there is no

Table continues on the next page...

SRX_GBL_STATUS field descriptions (continued)

Field	Description
	message in the DMA buffers for read and the software still reads the Slow Serial Message registers. Default value is 0. 0 No underflow 1 Underflow occurred
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.3.4 Fast Message Ready Status Register (SRX_FMSG_RDY)

The bits of this register are asserted when a Fast Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (viaFast Message DMA Control Register (SRX_FDMA_CTRL)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (viaFast Message Ready Interrupt Control Register (SRX_FRDY_IE)) and the channel is enabled, the message ready status bit in this register will be asserted on Fast Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Proper flow cannot be guaranteed if neither is selected.

These bits can be configured to be cleared via "Write 1 to Clear" or "Fast Clearing Mechanism". The configuration is done by writing to the FAST_CLR bit in theGlobal Control Register (SRX_GBL_CTRL) .

NOTE

> Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														F_ RDY	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FMSG_RDY field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 F_RDY	Fast Message Data Ready Status bits for channels 1 to 0 . Each bit indicates that a Fast Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that

Table continues on the next page...

SRX_FMSG_RDY field descriptions (continued)

Field	Description
	channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Fast Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). **NOTE:** These bits are either "write 1 to clear" (w1c) or self-clearing depending on the value of the FAST_CLR BIT in the Global Control Register (GBL_CTRL). The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See SectionGlobal Control Register (SRX_GBL_CTRL) for details on FAST_CLR bit. **NOTE:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 fast message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. **NOTE:** When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits. 0 No Fast Message is available to be read out via Interrupt 1 Fast Message is available to be read out via Interrupt

51.3.5 Slow Serial Message Ready Status Register (SRX_SMSG_RDY)

The bits of this register are asserted when a Slow Serial Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (viaFast Message DMA Control Register (SRX_FDMA_CTRL) andSlow Serial Message DMA Control Register (SRX_SDMA_CTRL)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (viaFast Message Ready Interrupt Control Register (SRX_FRDY_IE) andSlow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)) and the channel is enabled, the message ready status bit in this register will be asserted on Slow Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Erroneous flow can happen if neither is selected.

These bits can be configured to be cleared via "Write 1 to Clear" or "Fast Clearing Mechanism". The configuration is done by writing to the FAST_CLR bit in theGlobal Control Register (SRX_GBL_CTRL).

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														S_ RDY	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SMSG_RDY field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 S_RDY	Slow Serial Message Data Ready Status bits for channels 1 to 0 . Each bit indicates that a Slow Serial Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Serial Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). **NOTE:** These bits are either "write 1 to clear" (w1c) or self-clearing depending on the value of the FAST_CLR bit in the Global Control Register (GBL_CTRL). The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See SectionGlobal Control Register (SRX_GBL_CTRL) for details on FAST_CLR bit. **NOTE:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 Fast Message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. **NOTE:** When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits. 0 No Slow Serial Message is available to be read out via Interrupt 1 Slow Serial Message is available to be read out via Interrupt

51.3.6 Data Control Register 1 (SRX_DATA_CTRL1)

The Data Control Register 1 (DATA_CTRL1) is described below.

NOTE

Reset value changes depending on the number of channels that are enabled on the device.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	NIBBCH0			0	NIBBCH1			0	Reserved			0	Reserved		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	Reserved			0	Reserved			0	Reserved			0	Reserved		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DATA_CTRL1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 NIBBCH0	Number of Data Nibbles supported in Channel 0 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 NIBBCH1	Number of Data Nibbles supported in Channel 1 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 Reserved	This field is reserved.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 Reserved	This field is reserved.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 Reserved	This field is reserved.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 Reserved	This field is reserved.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 Reserved	This field is reserved.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved.

51.3.7 Fast Message DMA Control Register (SRX_FDMA_CTRL)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

RW

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														FDMA_EN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FDMA_CTRL field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FDMA_EN	Enable DMA for Fast Messages on channels 1 to 0 . These bits are writable when corresponding Fast Message Ready Interrupt Enable bits are set to 0. 0 DMA for Fast Messages is disabled 1 DMA for Fast Messages is enabled

51.3.8 Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														SDMA_EN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SDMA_CTRL field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRX_SDMA_CTRL field descriptions (continued)

Field	Description
30–31 SDMA_EN	Enable DMA for Slow Serial Messages on channels 1 to 0 . These bits are writable when corresponding Slow Serial Message Ready Interrupt Enable bits are set to 0. 0 DMA for Slow Serial Messages is disabled 1 DMA for Slow Serial Messages is enabled

51.3.9 Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														FRDY_IE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FRDY_IE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FRDY_IE	Enable for Fast Message Ready Interrupt on channels 1 to 0 . These bits are writable when corresponding DMA enable bits are set to 0 in the Fast Message DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Fast Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit is asserted in this register, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Fast Messages is disabled 1 Interrupt on reception of Fast Messages is enabled

51.3.10 Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														SRDY_IE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SRDY_IE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 SRDY_IE	Enable for Slow Serial Message Ready Interrupt on channels 1 to 0 . These bits are writable when corresponding DMA enable bits are set to 0 in the Slow Serial DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Slow Serial Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit in this register is asserted, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Slow Serial Messages is disabled 1 Interrupt on reception of Slow Serial Messages is enabled

51.3.11 DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_DATA) via DMA. When FIFO is enabled, the DMA must read the complete message using this register only. Three read accesses must be made to read complete message.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHNUM				SCNIB				DNIB1				DNIB2				DNIB3				DNIB4				DNIB5				DNIB6			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_DATA field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 1 .
4–7 SCNIB	Status and Communication Nibble of message
8–11 DNIB1	Data Nibble 1. Always supported
12–15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16–19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20–23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24–27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28–31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

51.3.12 DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_CRC) via DMA. When FIFO is enabled, fast messages are read out from FIFO through the DMA Fast Message Data Read register only. Reading this register will generate an access error.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CRC4b				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_CRC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CRC4b	4-bit CRC value of the message.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.3.13 DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in theGlobal Control Register (SRX_GBL_CTRL) register. DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_TS) via DMA. When FIFO is enabled, fast messages are read out from FIFO through the DMA Fast Message Data Read register only. Reading this register will generate an access error.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

51.3.14 DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)

The above register (DMA_SMSG_BIT3) is common for all types of slow serial messages. Bit C defines the type of enhanced packet and bit TYPE defines whether it is short serial or enhanced serial message. Since this register is used to read both Short and Enhanced Serial Messages, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown inTable 51-2,Table 51-3 andTable 51-4.Table 51-2 andTable 51-3 shows the enhanced serial messages format to be used when reading this register for these two types of messages.Table 51-4 shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

NOTE

> If the TYPE bit in the register reads '0', then remainder bits (for CFG, ID and DATA) will be redundant and read zeros. The short serial message will be located in DMA_SMSG_BIT2 register fields. If TYPE bit is '1', then the message in register is Enhanced Serial Message and using CFG bit, software can decode the ID and DATA fields.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					CFG	ID7_4_ID3_0				0	ID3_0_DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_BIT3 field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 1 .
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message
5–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22–25 ID7_4_ID3_0	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 ID3_0_DATA15_12	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.3.15 DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)

This register (DMA_SMSG_BIT2) is common for all types of slow serial messages. Hence, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown inTable 51-2,Table 51-3 andTable 51-4.Table 51-2, andTable 51-3 shows the enhanced serial messages format to be used when reading this register for these two types of messages.Table 51-4 shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SMCRC						0				DATA[11:0]											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_BIT2 field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 SMCRC	6-bit CRC value of the message. When TYPE bit inDMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA[11:0]	12-bit Data Value

51.3.16 DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in theGlobal Control Register (SRX_GBL_CTRL) register.

DMA Source Size must be set to 32 bits when reading this register (DMA_SMSG_TS) via DMA.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

51.3.17 Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)

NOTE

> The value of the CH is device-specific. See the device configuration section for details.

The SENT Receiver module generates the Receiver Sensor Clock (or simply Receiver Clock) internally to sample the incoming filtered sensor data. The user software must program the prescaler value (for Receiver Clock) to match the ideal Sensor Tx Clock period. The generated Receiver Clock is compensated for variations and clock drift that might occur in the Sensor Tx Clock. SeeAdjustment for variation in sensor (Tx) clock for more details.

This register configures the generation and subsequent compensation logic of the Receiver Clock. User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in theChannel Enable Register (SRX_CHNL_EN) .

Address: 0h base + 60h offset + (16d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CM_PRSC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_EN	0	PRSC													
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_CLK_CTRL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 CM_PRSC	Compensated Prescaler Value. This is the compensated prescaler value calculated by Clock Compensation logic and this value is used to compensate for sensor Tx clock variation. This value is indicated as a read only field for use by user software, if needed. This is a 15-bit value. '= Rx Prescaler Value' No variation, hence no compensation or Compensation is disabled '< Rx Prescaler Value' Tx Clock is SLOWER than the programmed receive clock '> Rx Prescaler Value' Tx Clock is FASTER than the programmed receive clock Thus, the new Receiver Clock Tick Period can be computed by software as

Table continues on the next page...

SRX_CH*n*_CLK_CTRL field descriptions (continued)

Field	Description
	Compensated Prescaler Value x High Frequency Receiver Clock Period Note: The value provided in this field is ceil value of actual tick period as measured by the Receiver.
16 COMP_EN	Compensation Enable. To enable compensation logic to adjust the receiver clock against variation in Tx clock on this channel. User must set the Prescaler Value before enabling this bit. 0 Compensation is disabled 1 Compensation is enabled
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 PRSC	Rx Prescaler Value. This factor is required to generate required Receiver Clock tick from the High Frequency Receiver Clock. This value can only be written when Channel Enable bit is Zero. Hence this value must be programmed before enabling the channel. If required receiver clock tick period (Trx_clk) is not divisible by the high frequency receiver clock (protocol clock) period (Thf_clk), the required prescaler value will have a fractional part which cannot be programmed. In this case then user is advised to program the prescaler value to the lower integer value. 0 Prescaler is bypassed. Should not be used. Non-Zero Value Enables Prescaler The required value is computed by software as follows: $\text{Prescaler\_value} = \text{floor}\left(\frac{T_{\text{rx\_clk}}}{T_{\text{hf\_clk}}}\right)$ **Equation 54** (both clock periods taken in µs). Please seeHigh frequency receiver clock (protocol clock) requirements before setting this value. e.g. if the required receiver clock tick period is 43 µs and the High Frequency Receiver clock is selected to be 62.5 MHz (or 0.016 µs) then, Prescaler Value = 43/0.016 = 2687.5. Hence user must program 2687 in this register.

51.3.18 Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)

This registers stores information about certain events and error conditions that occur in the channel during the message reception process. The assertion of these bit is not linked to any message but it is a general alert to the CPU that a particular condition (error or normal event) happened in the recent past. Messages are automatically discarded when an error is detected.

User software can clear these bit at any time by writing 1 to them. This register can be updated at any time irrespective of the setting in theChannel Enable Register (SRX_CHNL_EN) bit for the corresponding channel.

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 64h offset + (16d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUS_IDLE	0			CAL_RESYNC	CAL_20_25	SMSG_OFLW	FMSG_OFLW	0	PP_DIAG_ERR	CAL_LEN_ERR	CAL_DIAG_ERR	NIB_VAL_ERR	SMSG_CRC_ERR	FMSG_CRC_ERR	NUM_EDGES_ERR
W	w1c				w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_STATUS field descriptions

Field	Description
0 BUS_IDLE	Bus Idle Status. This bit indicates that the sensor interface has been idle for more than the period defined by the programmed Bus Idle Count value(inChannel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)). CPU should write 1 to this clear this bit once it is read. 0 Bus is not idle 1 Channel has been idle for more than the allowed value
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 CAL_RESYNC	Successive Calibration Check (Option 2 of SAE SENT Spec) Resynchronized. This bit indicates that Successive Calibration Check (Option 2 of SAE Spec) has failed three times and has resynchronized to

Table continues on the next page...

SRX_CHn_STATUS field descriptions (continued)

Field	Description
	assume 3rd message to be correct and will be received. The successive calibration check error will also be asserted during re-synchronization (to indicate the 3rd error being detected). 0 No interrupt 1 Interrupt Status condition has occurred
5 CAL_20_25	Calibration Variation 20 - 25% Interrupt Status. This bit indicates that the calibration pulse received on this channel has variation in between ±20% to ±25% from 56 ticks. 0 No Interrupt 1 Interrupt Status condition has occurred
6 SMSG_OFLW	Slow Serial Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow will cause state machine to halt stopping reception of new messages. 0 No Interrupt 1 Interrupt Status condition has occurred
7 FMSG_OFLW	Fast Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow happens when effective data read rate by CPU is slower than data receive rate on channel. On overflow, new incoming messages will overwrite the previous messages (that are not read out) stored in the receiver buffers. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 No Interrupt 1 Interrupt Status condition has occurred
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 PP_DIAG_ERR	This diagnostic checks status bit indicates that the ratio of calibration pulse length to overall message length (with pause pulse) is more than ±1.5625% between two messages. This check is valid only for messages with pause pulse. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
10 CAL_LEN_ERR	This diagnostic checks status bit indicates that Calibration pulse is more than 56 ticks ±25%. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
11 CAL_DIAG_ERR	Successive Calibration pulses differ by ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the frequency of protocol clock (SeeHigh frequency receiver clock (protocol clock) requirements). This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
12 NIB_VAL_ERR	Any nibble data value <0 or >15. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
13 SMSG_CRC_ ERR	Checksum error in Slow Serial Message. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it.

Table continues on the next page...

SRX_CH*n*_STATUS field descriptions (continued)

Field	Description
	0 Error Check has passed 1 Error Check has failed
14 FMSG_CRC_ERR	Checksum error in Fast Message. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
15 NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel. This bit will not set in case of "Option 2" for successive calibration pulse check is selected. This error will only set when "Option 1" for successive calibration check is selected (SUCC_CAL_CHK). This bit is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.3.19 Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained inChannel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS) and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in theChannel Enable Register (SRX_CHNL_EN) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 68h offset + (16d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BUS_IDLE_CNT				IE_CAL_RESYNC	IE_CAL_20_25	IE_SMSG_OFLW	IE_FMSG_OFLW	FCRC_CHK_OFF	IE_PP_DIAG_ERR	IE_CAL_LEN_ERR	IE_CAL_DIAG_ERR	IE_NIB_VAL_ERR	IE_SMSG_CRC_ERR	IE_FMSG_CRC_ERR	IE_NUM_EDGES_ERR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DCHNG_INT	CAL_RNG	PP_CHKSEL	FCRC_TYPE	FCRC_SC_EN	SCRC_TYPE	PAUSE_EN	SUCC_CAL_CHK	FIL_CNT							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

SRX_CH*n*_CONFIG field descriptions

Field	Description
0–3 BUS_IDLE_CNT	Bus Idle Count. This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000 Disabled. 0001 127*2 Receiver Clock Tick Counts 0010 127*4 Receiver Clock Tick Counts 0100 127*8 Receiver Clock Tick Counts 1000 127*16 Receiver Clock Tick Counts
4 IE_CAL_ RESYNC	Successive Calibration Check Resynchronized Interrupt Enable: This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of "Option 2" being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 Interrupt is disabled 1 Interrupt is enabled
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable. This bit enables interrupt assertion when corresponding status bit is set. 0 Interrupt is disabled 1 Interrupt is enabled
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Slow Serial Messages in corresponding channel 0 Interrupt is disabled 1 Interrupt is enabled
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Fast Messages in corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
8 FCRC_CHK_ OFF	Fast Message CRC Check Off: This bit can be used to switch off CRC check in Fast Message 0 Check is enabled 1 Check is disabled/off
9 IE_PP_DIAG_ ERR	Ratio of calibration pulse length to message length varies by more than ±1.5625% between two frames. Valid for messages with pause pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

SRX_CH*n*_CONFIG field descriptions (continued)

Field	Description
10 IE_CAL_LEN_ERR	Calibration pulse is wider than 56 ticks ±25%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
11 IE_CAL_DIAG_ERR	Successive Calibration pulses differ by more than ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
12 IE_NIB_VAL_ERR	Any nibble data value <0 or >15. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
13 IE_SMSG_CRC_ERR	Checksum error in Slow Serial Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
14 IE_FMSG_CRC_ERR	Checksum error in Fast Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
15 IE_NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data: This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, or FIFO, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 All Fast Messages will be received even if data nibbles are same 1 Only Fast Messages with differing values of data nibbles will be received
17 CAL_RNG	Valid Calibration Pulse Range Selection: This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 20% variation is acceptable 1 25% variation is acceptable
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection: This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run 1 Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.

Table continues on the next page...

SRX_CH*n*_CONFIG field descriptions (continued)

Field	Description
19 FCRC_TYPE	Fast Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble is XORed)
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable. This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast Messages. 0 Status and Communication Nibble not included in CRC calculation 1 Status and Communication Nibble is included in CRC calculation
21 SCRC_TYPE	Slow Serial Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble XORed)
22 PAUSE_EN	Pause Pulse Enable. Enables the channel receiver to detect a pause pulse. 0 Detection of Pause Pulse is disabled 1 Detection of Pause Pulse enabled
23 SUCC_CAL_CHK	Successive Calibration Pulse Check Method. This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 Option 2 i.e. Low Latency Option as per SAE Specification 1 Option 1 i.e. Preferred but High Latency Option as per SAE Specification
24–31 FIL_CNT	Input Filter Sample Count. This indicates twice the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'. 0 No filtering. Input from device's pad is only synchronized 1 Non-Zero Filtering is enabled

51.3.20 Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_DATA)

Address offset: 0x0160 + 0x18 x n

This is register CHn_FMSG_DATA. The contents of this register are same asDMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 160h offset + (24d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHNUM				SCNIB				DNIB1				DNIB2				DNIB3				DNIB4				DNIB5				DNIB6			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_FMSG_DATA field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 1 .
4–7 SCNIB	Status and Communication Nibble of message
8–11 DNIB1	Data Nibble 1. Always supported
12–15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16–19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20–23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24–27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28–31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

51.3.21 Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_CRC)

Address offset: 0x0164 + 0x18 x n

This is register CHn_FMSG_CRC. The contents of this register are same asDMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC) .

Address: 0h base + 164h offset + (24d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CRC4b				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_FMSG_CRC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRX_CH*n*_FMSG_CRC field descriptions (continued)

Field	Description
12–15 CRC4b	4-bit CRC value of the message.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.3.22 Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_TS)

Address offset: 0x0168 + 0x18 x n

This is register CHn_FMSG_TS. The contents of this register are same as DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS).

Address: 0h base + 168h offset + (24d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_FMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

51.3.23 Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH*n*_SMSG_BIT3)

Address offset: 0x016C + 0x18 x n

This is register CHn_SMSG_BIT3. The contents of this register are same asDMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 16Ch offset + (24d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					CFG	ID7_4_ID3_0				0	ID3_0_DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_BIT3 field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 1 .
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message
5–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22–25 ID7_4_ID3_0	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 ID3_0_DATA15_12	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.3.24 Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH*n*_SMSG_BIT2)

Address offset: 0x0170 + 0x18 x n

This is register CHn_SMSG_BIT2. The contents of this register are same asDMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2) .

Address: 0h base + 170h offset + (24d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SMCRC						0				DATA[11:0]											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_BIT2 field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 SMCRC	6-bit CRC value of the message. When TYPE bit inDMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA[11:0]	12-bit Data Value

51.3.25 Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH*n*_SMSG_TS)

Address offset: 0x0174 + 0x18 x n

This is register CHn_SMSG_TS. The contents of this register are same as DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS).

Address: 0h base + 174h offset + (24d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

51.4 Functional description

This section describes the function of the SRX.

51.4.1 Initialization sequence

All the channels are required to be configured before enabling them by writing into corresponding channel enable bit in the Channel Enable Register (SRX_CHNL_EN). There are some registers which must be configured and some which are to be changed if reset value is not as required for that channel. After enabling the channel, no configuration should be changed and if required to be changed, the user needs to first disable the channel and then reconfigure the channel parameters. The following must to be configured before enabling the corresponding channel:

- Time Stamp Prescaler Value in Global Control Register (SRX_GBL_CTRL) to generate the time stamping clock
- Rx Prescaler Value in Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL) togenerate the channel's receiver clock
- Compensation enable in Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)
- Channel Configuration settings in the Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)
- Number of Data Nibbles in Channel in the Data Control Register 1 (SRX_DATA_CTRL1)
- Control bits that control whether messages received in channel are to read by interrupt or DMA in Fast Message DMA Control Register (SRX_FDMA_CTRL), Slow Serial Message DMA Control Register (SRX_SDMA_CTRL), Fast Message Ready Interrupt Control Register (SRX_FRDY_IE) and Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)

51.4.2 DMA read logic

The SENT Receiver module generates separate DMA requests for all Fast Messages and Slow Serial Messages received. The DMA requests are common for all channels. Each packet stored in the channels' buffers are appended with a time stamp taken from a free running time keeping counter and the channel number on which it was received. Details on this time stamp counter can be found in Time stamp logic.

51.4.2.1 DMA request for Fast Message reading via FIFO

A single DMA request is generated for all Fast Messages received on any channel. There is a DMA read FIFO to store the incoming messages from all the channels and allow them to be read via DMA. The DMA operation will happen as follows:

- DMA request will be asserted when number of messages stored in FIFO crosses the configured watermark level (in Global Control Register (GBL_CTRL)).
- DMA request will be de-asserted when number of messages stored goes below watermark level. It is assumed that DMA Controller's TCD would be set to transfer bytes equal to the messages indicated by watermark level so as to empty the FIFO.
- It is assumed that when responding to a DMA request, the DMA controller will read at a rate faster than the rate at which messages are getting received and stored in FIFO.
- The complete received message can be read by three consecutive read accesses to DMA Fast Message Data Read Register. The message will be read out in following sequence:
    - Fast Message data as described by DMA Fast Message Data Read Register (DMA_FMSG_DATA)
    - CRC for the Fast Message as described by DMA Fast Message CRC Read Register (DMA_FMSG_CRC)
    - Time stamp value for the Fast Message as described by DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)
- The channel number is prefixed in each valid message (no error) to allow software to identify messages pertaining to a channel
- Table 51-1 shows the format in which the messages will be stored in any memory-mapped message buffer and when being read out from FIFO.
- Since multiple messages can be received at the same time, there is a Fixed round robin sequencing logic, which will scan all active channels configured for DMA and store their message into the FIFO one by one. Channel configured for DMA but do not have ready messages will be automatically skipped till their turn comes again in the round robin sequence.
- Messages received can be stored in a different order to which they are actually received. The time stamp will indicate the order of reception of messages in a particular channel. However, the time stamp cannot be used to determine the order across channels.

This figure below shows the data for the three 32-bit accesses to be made by DMA to get one complete message with time stamp.

Table 51-1. Format of storing of Fast Messages in FIFO

CH NUM [3:0]	Status and Comm [3:0]	Data 1 [3:0]	Data 2 [3:0]	Data 3 [3:0]	Data 4 [3:0]	Data 5 [3:0]	Data 6 [3:0]
			CRC [3:0]				
TIME STAMP [31:0]							

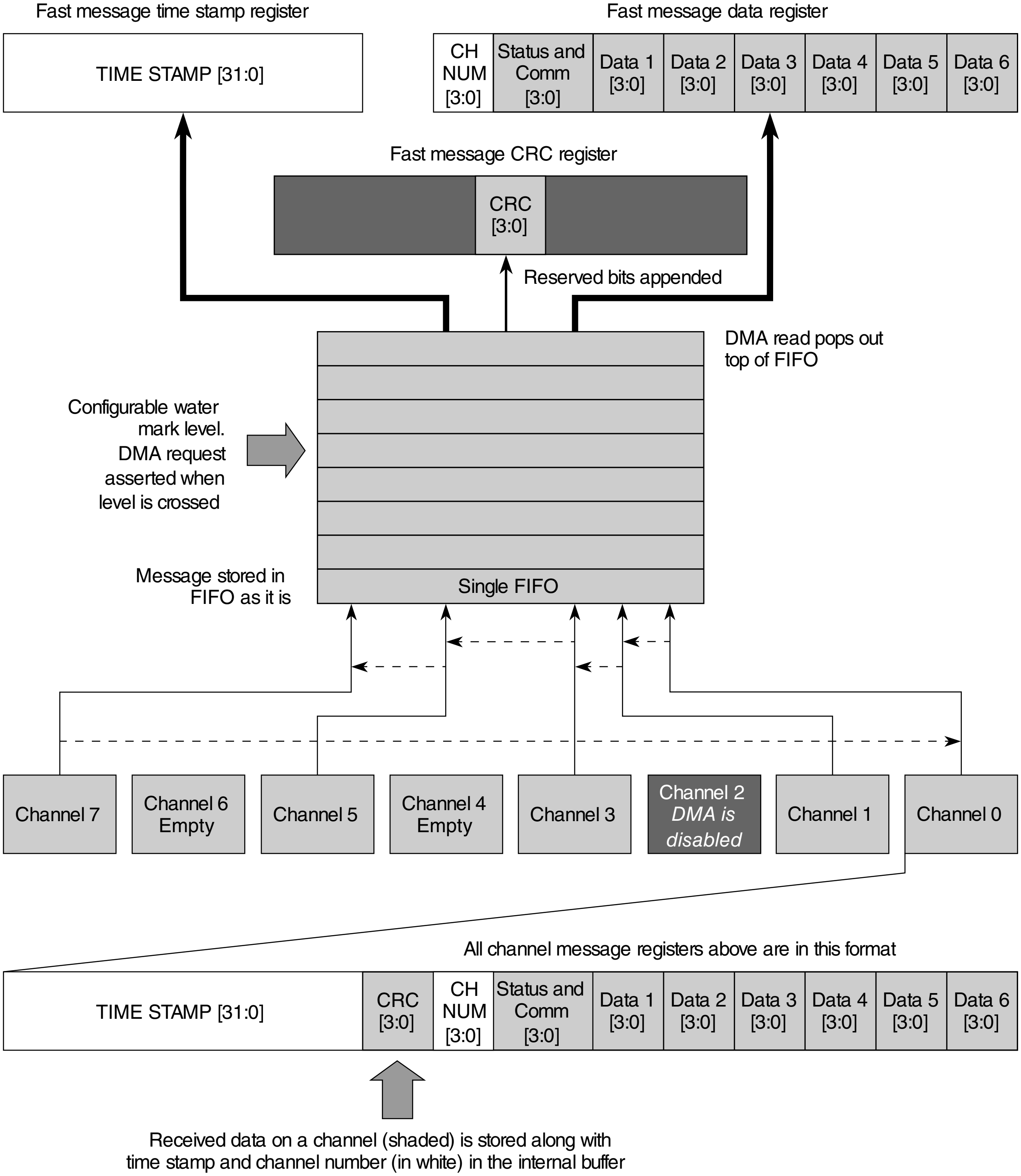

Fast message time stamp register
TIME STAMP [31:0]
Fast message data register
CH NUM [3:0]
Status and Comm [3:0]
Data 1 [3:0]
Data 2 [3:0]
Data 3 [3:0]
Data 4 [3:0]
Data 5 [3:0]
Data 6 [3:0]
Fast message CRC register
CRC [3:0]
Reserved bits appended
DMA read pops out top of FIFO
Configurable water mark level. DMA request asserted when level is crossed
Message stored in FIFO as it is
Single FIFO
Channel 7
Channel 6 Empty
Channel 5
Channel 4 Empty
Channel 3
Channel 2 DMA is disabled
Channel 1
Channel 0
All channel message registers above are in this format
TIME STAMP [31:0]
CRC [3:0]
CH NUM [3:0]
Status and Comm [3:0]
Data 1 [3:0]
Data 2 [3:0]
Data 3 [3:0]
Data 4 [3:0]
Data 5 [3:0]
Data 6 [3:0]
Received data on a channel (shaded) is stored along with time stamp and channel number (in white) in the internal buffer

Figure 51-2. Fast Message DMA Read Logic using FIFO

51.4.2.1.1 Fixed round robin sequencing logic

Figure 51-2 shows a scenario where 8 channels (0-7) with all channels are enabled for DMA reading except channel 2. DMA request will be asserted when FIFO has received packets more than watermark level set. Figure 51-2 shows channels 6 and 4 have not received the complete message. The following points describe the complete round robin sequencing scheme.

- The round robin sequencing logic which stores the messages in the FIFO by looping through each channels' Fast Message register and checking for the availability of the message.
- As shown in Figure 51-2, the arbitration logic will check for messages in sequence 0 -> 1 -> 3 -> 5 -> 7 and back to channel 0 and keep looping. Channel 2 is skipped as it's not enabled to be read via DMA) and channel 4 and 6 are skipped as message is not received completely at the instance the sequencing logic checked their buffers. This sequence continues in this fashion.
- The sequence in which messages are stored in FIFO is not deterministic and depends on the sequence they are detected after the noise filtering by the input filter of the channel.
- The DMA controller responds to the DMA request by reading from the DMA Fast Message Read register. When a DMA read happens, a message is read from the FIFO and split into three 32-bit values and provided to DMA in the order as described in previous section. The three 32-bit values together form one complete Fast Message.
- Once all three 32-bit access are completed, the next data from the FIFO is popped.

Figure 51-2 also shows how the received data (light gray colored boxes) from a channel is stored in FIFO along with the time stamp and channel number appended (both white colored).

51.4.2.2 DMA request for Fast Message reading when FIFO is disabled

In this mode no FIFO will be used as it is disabled via the FIFO Enable bit in the Global Control Register (GBL_CTRL). Fast Messages received are stored in a single system bus clock domain buffer to be read out by the DMA Controller via the DMA Fast Message Read register set. Note the DMA Controller must read all 3 DMA Read Registers to obtain the complete message, unlike the case when FIFO is enabled and message is read out from the DMA Fast Message Data Read register only. The data flow will happen as follows:

- A DMA request will be asserted when any of the DMA enabled channels has successfully received one complete Fast Message. The request is kept asserted till all messages (across DMA enabled channels) have been read out.
- DMA Controller should read the complete message by accessing each register of the DMA Fast Message Read register set, the registers being:
  - DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)
  - DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)
  - DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)
- A fixed round robin sequencing runs, as shown in Figure 51-3, and this logic checks for messages in sequence 0→1→3 →5→7 and back to channel 0 and so on. Channel 2 is skipped as it is not DMA enabled) and channel 4 and 6 are skipped as message is not received completely. This sequence continues like this.
- The sequence in which messages are read through DMA interface is not deterministic and depends on the sequence in which they are completely received.
- When the sequencing logic detects a message ready to be read, it will update the DMA Fast Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 read accesses to the above mentioned registers)
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- User software should ensure there is no overflow in channels. In case overflow occurs, the message in the channel buffer will get overwritten by the newly received message. However, when a DMA read is in progress, the buffers will not be overwritten if overflow occurs to maintain data integrity.
- The Round Robin sequencing logic loops through each DMA enabled channel and halts at any channel having data till it is read out by the DMA Controller.

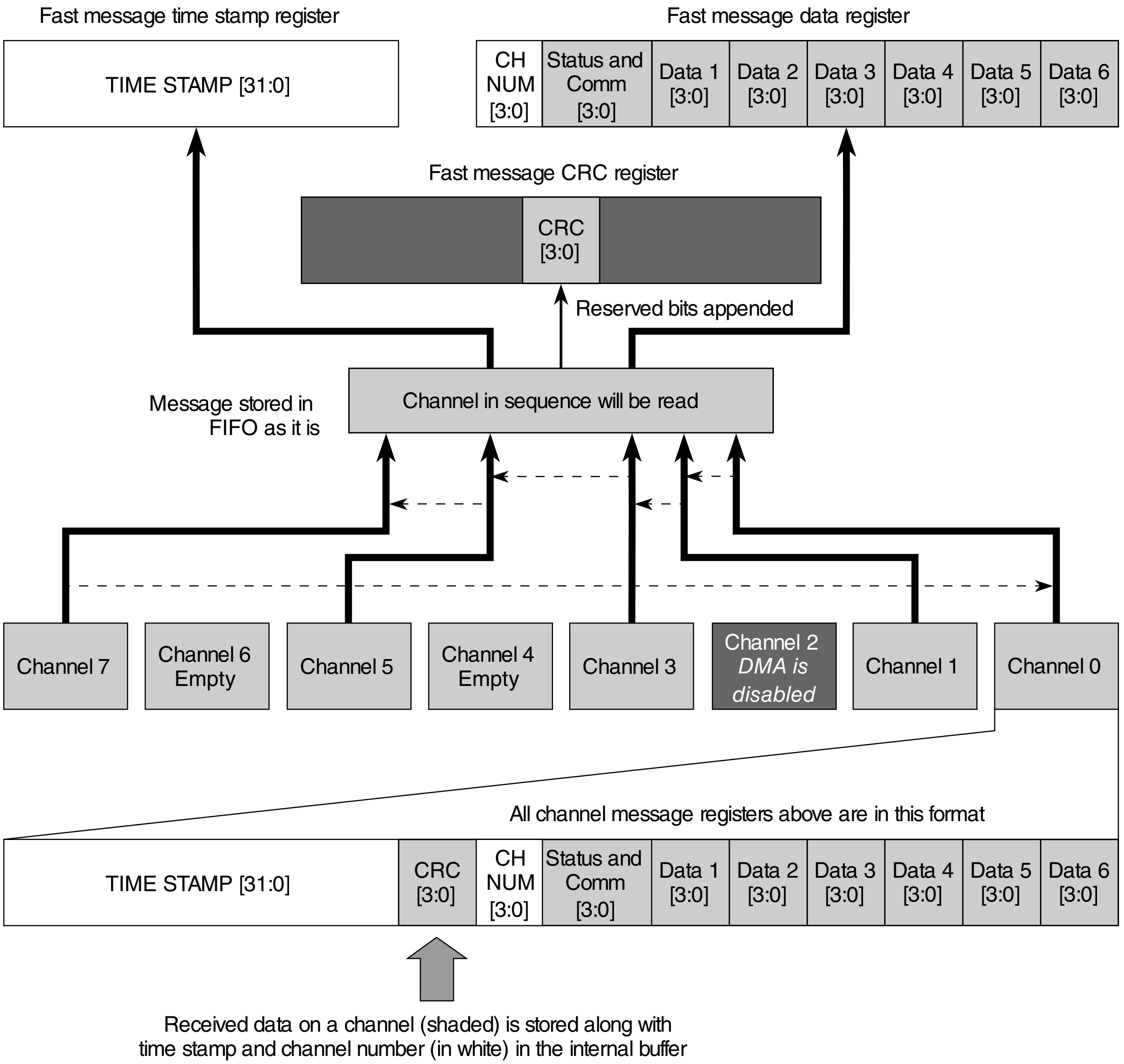


Figure 51-3. Fast Message DMA read logic (when FIFO is disabled)

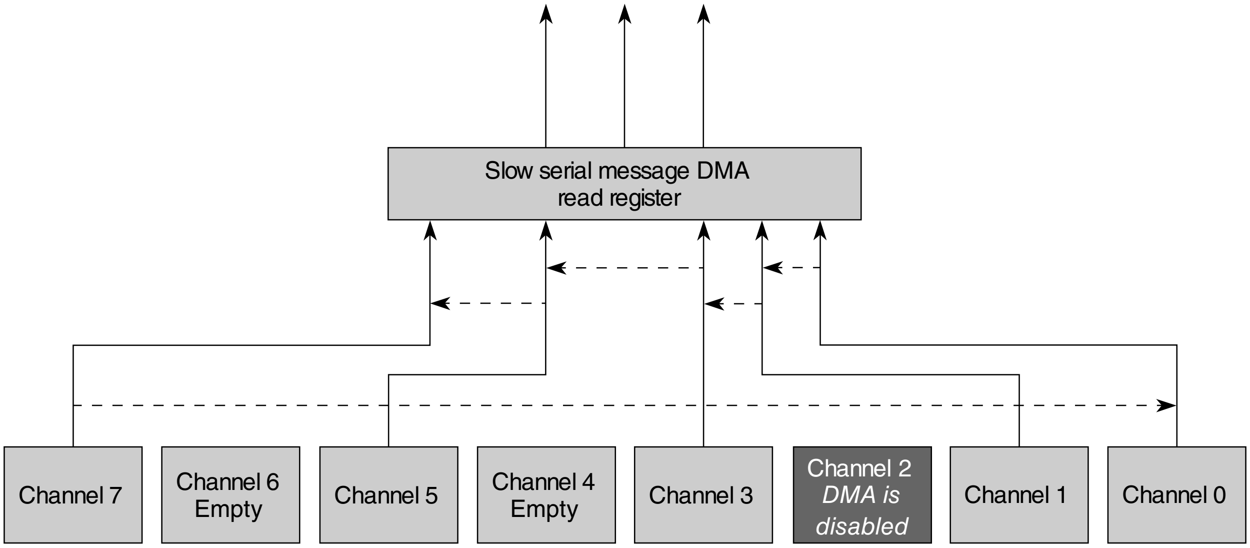
51.4.2.3 DMA request for Slow Serial Message reading

A separate DMA request (different from DMA request for Fast Message) is generated for all slow serial messages received on all channels. The data flow will happen as follows:

- Request will be asserted when any of the DMA enabled channels has successfully received one complete serial message (short or enhanced) and kept asserted till all messages have been read from their respective memory-mapped registers

- Similar to DMA read for Fast Messages, data for slow serial messages will be read from DMA Slow Serial Message Read register set in memory map, the register set comprising of:
    - DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)
    - DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)
    - DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)
- A fixed round robin sequencing logic will be running and it will update the DMA Slow Serial Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 DMA read accesses to the above mentioned registers)
- The channel number and time stamp will also be appended for every message
- A message type bit (TYPE) will also be appended to distinguish between short and enhanced serial messages
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- Since serial messages receive rate is very slow, user software should ensure there is no overflow
- The fixed round robin sequence logic for DMA read is the same as that used for Fast Messages (see DMA request for Fast Message reading when FIFO is disabled)

Figure 51-4 shows the sequencing of channels for serial messages to be read when DMA read is done on DMA Slow Serial Message Read register set.



Note: The arrows in the figure indicate next DMA transfer. One DMA Read Register set (3 registers) will be read in

Figure 51-4. Slow Serial Message DMA read logic

The formats in which the 3 types of Serial Messages will be stored in the buffers are shown in figures below.

Table 51-2. Enhanced Serial Message with 8-bit ID field

31	1		16	15		0				0
CH NUM [3:0]	P				0	C	ID [7:4]	0	ID [3:0]	0
			CRC [5:0]		Data Field [11:0]					
TIME STAMP [31:0]										

Table 51-3. Enhanced Serial Message with 4-bit ID field

31	1		16	15		1				0
CH NUM [3:0]	P				0	C	ID [3:0]	0	Data Field [15:12]	0
			CRC [5:0]		Data Field [11:0]					
TIME STAMP [31:0]										

Table 51-2 and Table 51-3 shows the enhanced serial messages. Bit C (shown in figure) or CFG (as shown in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) description) defines the type of enhanced packet and bit P (shown in figure) or bit TYPE (shown in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) description) defines whether it is short serial or enhanced serial message. CRC, ID and Data fields are stored in format as received and shown in SAE specification with just one exception that CRC is shifted to align with 16-bit boundary.

Table 51-4 shows the short serial message. Field locations have been placed in such a way to allow a common register to read all the slow serial messages and also match the bit positions in actual message defined by SAE Specification.

Table 51-4. Short serial message

31	0		16		15	0
CH NUM [3:0]	P					
			CRC [3:0]		ID [3:0]	Data Field [7:0]
TIME STAMP [31:0]						

Again, three 32-bit accesses to be made by DMA to get one complete message with time stamp.

51.4.3 Message reading via interrupts

There is one interrupt request per channel for a Fast message received and one for a serial message received. Each channel is individually configurable to enable or disable interrupt on reception of Fast or Slow Serial messages separately. In case of interrupt request of Fast Message, an interrupt will be asserted when any of the channels has received a complete Fast Message without errors and that channel is not configured for DMA Reads. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing all channels that are enabled to give out messages via interrupts through the Register Interface.

In response to these interrupts, the CPU is expected to read interrupt status register for fast or Slow Messages (depending on which interrupt was asserted) and store it in system memory. Using this status register value, the interrupt routine can decide from which channels the message is to be read. For instance if bit corresponding to channel 0 is set then routine should read from the address location corresponding to Channel 0 Fast Message Read registers and move onto the next asserted bit. The interrupt routine should scan through all the bits of this stored interrupt status register value and read corresponding channel's fast or Slow Message registers. In this manner all available messages can be read. After scanning through all the bits of status register stored in system memory, routine can read again the message ready interrupt status register to find out if any other channel has received in the mean time. This will save another interrupt routine time mainly from stacking/unstacking delay.

It is assumed that the interrupt controller will be able to respond fast enough to avoid any overflow as only one message per channel will be stored and the system bus clock frequency is sufficient to avoid this.

51.4.3.1 Suggested software interrupt service routine steps

These are suggested step to have an interrupt routine that can read all messages quickly and avoid any underflow/overflow. Actual implementations can vary.

When separate Fast Message and Slow Serial Message Interrupts are used:

The Fast Message Interrupt should have higher priority than Slow Serial Message Interrupt. The suggested ISR steps for a Fast Message Interrupts can be

- Read the Fast Message Ready Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register set (3 registers). Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Fast Message Status Registers again to ensure no new message has arrived during the ISR.
- If there are no bits set, exit the ISR, else repeat this process.

The ISR steps for Slow Serial Message Interrupt can be same as for Fast Message ISR.

When common Fast Message and Slow Serial Message Interrupts are used:

The suggested ISR steps can be

- Read the Fast Message Status Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Slow Serial Message Status Register into a new variable

- Shift out bits starting from MSB from this new variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Slow Serial Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, software can again check both Fast and Slow Serial Message Ready register to ready any message that might have been received while the routine was being serviced, or can choose to exit the routine.

51.4.4 Overflow behavior

This section describes the overflow behavior.

51.4.4.1 Fast messages buffers/FIFO

Overflow in Fast Message buffers or FIFO can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full. When FIFO is used, overflow is indicated when the FIFO gets full and another fast message is to be stored.

When an overflow event occurs, the newly received message overwrite the previously stored message and the overflow status bit is set. However, if overflow asserts while CPU/DMA is reading that channel (i.e. all 3 read registers have not yet been read), then that message is not overwritten. While messages are being overwritten, the slow message reception is not hampered.

NOTE

> In the case of a fast message overflow and continuous reading of register SRX_CHn_FMSG_DATA, SRX_CHn_FMSG_CRC, and SRX_CHn_FMSG_TS without clearing the SRX_FMSG_RDY[F_RDYn] bit there is a possibility that one message will be lost. Additionally, if the pause pulse feature is enabled the module asserts up to two NUM_EDGES_ERR in the status register (SRX_CHn_STATUS). In this case up to two frames can be lost. Note that some debuggers perform a continuous read of memory which can cause this issue to occur.

In order to avoid this software should ensure that fast message overflow does not occur. If interrupts are used to read the SENT messages the interrupt for data reception should be enabled for every channel n (SRX_FRDY_IE[FRDY_IEn]=1) and the interrupt priority should be such that the software is able to read the message before the next message arrives. When using Direct Memory Accesses (eDMA) to access the messages the DMA request from the SENT module should be serviced before the next message arrives. The minimum duration between the reception of two consecutive messages in one channel is 92 times the utick length (time).

Else ensure that the SRX_CHn_FMSG_DATA, SRX_CHn_FMSG_CRC, and SRX_CHn_FMSG_TS registers are not read continuously either in the software code or as a result of a debugger being connected. They should be read once per message and the FMSG_RDY[F_RDYn] bit should be cleared after the reads.

51.4.4.2 Slow message buffers

Overflow in Slow Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, further reception of slow messages is halted until the buffers are read out by CPU or DMA. Since it is less likely that CPU or DMA do not read the read registers for 32 or 36 fast message length duration, an overflow in slow messages would not be as frequent as in fast messages.

51.4.5 Adjustment for variation in sensor (Tx) clock

The clock used by the Channel Receiver (referred to as Receiver Clock, or Rx_CLK) to sample the message nibbles is generated from a High Frequency Receiver Clock or Protocol Clock. This high frequency receiver clock is divided using a prescaler counter which generates a tick when it rollovers on reaching a count equal to the Prescaler Value programmed in the channel's clock control register. The receive clock is generated separately for each channel. By varying this Prescaler Factor, we can control the period of the receive clock and make it close to the Transmitter clock used by the sensor for the particular channel.

After reset and once channel is enabled, the compensated prescaler value is set to Zero and the user software programs the Prescaler Value to obtain the desired receiver clock frequency for the particular channel. Thus, the receiver starts working by assuming that the transmitter is sending messages on the frequency that is programmed in its channel clock control register. The channel's input after filtering is continuously measured by the Prescaler Counter. The compensation logic checks each pulse length and determines the calibration pulse from them. When a calibration pulse is detected, the compensated prescaler value is computed and stored in the Channel's Clock Control Register (Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)). Software can use this value to determine the variation in Tx Clock.

Note

> A pause pulse may incidentally be detected as a synchronization/calibration pulse during start up after reset. The correction value would be computed on the pause pulse too but since the next synchronization pulse would follow, the logic would again detect the falling edge on the synchronization pulse and the correction value would now be correctly computed on the correct synchronization pulse.

The Clock Compensation or Correction happens for every messages that is received and the compensated prescaler value remains constant for that message.

51.4.5.1 Adjustment for nibble length variation

Section 6.2 of SAE Specifications specifies the maximum allowable limits for clock drift and jitter for the transmitter and receiver separately. This can cause the length of nibbles to vary and can cause incorrect sampling of data nibbles. This variation is adjusted by the SENT Receiver when nibbles are being sampled.

51.4.6 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value (i.e. sensor input) is validated and latched in the filtered output flop (also running on high frequency receiver clock). If the opposite edge appears on the sensor input before validation (i.e. overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG). When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register (Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)). A timing diagram of the input filter when FIL_CNT is 4, is shown in the following figure.

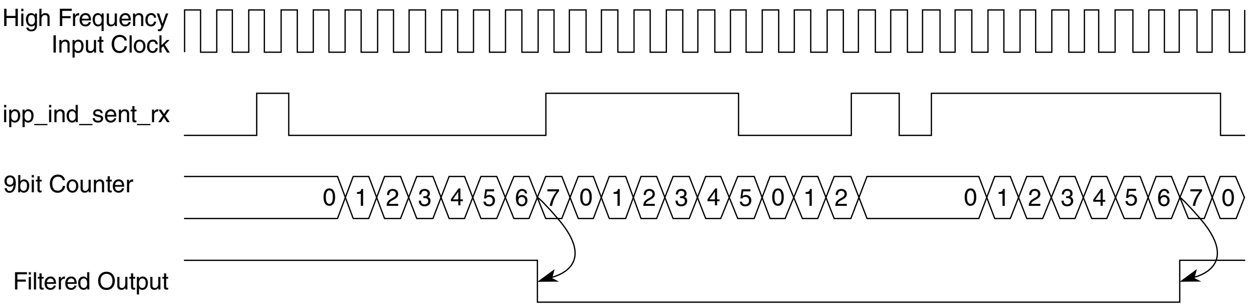


Figure 51-5. Input programmable filter timing diagram

51.4.7 Receiver diagnostics

Most of the receiver diagnostics are controlled by the Fast Message State Machine. The following checks are conducted by the receiver in each channel

- Calibration pulse length < 56 clock ticks – 25% or > 56 clock ticks + 25%
- Not the expected number of falling edges between calibration pulses. (Message length is pre-defined by each sensor device and programmed by user for each channel)
- Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or Recommended.
- Any nibble data values measured as < 0 or > 15

- Successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used.
- For messages with pause pulse, an additional diagnostic is done; however, this diagnostic will not fail any message (to reduce latency in message reception) but the failure of this diagnostic will be indicated in the channel's status register.

On detection of any of the above errors, the receiver rejects the current packet until a new calibration pulse is detected. Occurrence of an error is flagged in the respective channel's status register (Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)). If error interrupt is enabled, an error interrupt will be generated. There is a single error interrupt generated for all errors, so the CPU must read the status register to find out which error occurred. Once error condition is flagged, subsequent errors (except NUM_EDGES_ERR) are automatically masked out until a valid calibration pulse is detected, after which all checks are automatically enabled. This is done to avoid multiple assertions of the interrupt to CPU in case the channel become unstable and send wrong data continuously.

51.4.7.1 Calibration pulse length check

The calibration pulse length check happens in conjunction with Clock Compensation Logic since current calibration pulse is detected in this block. The Clock Compensation Logic detects a calibration pulse of length between 42 and 70 tick counts. This check will assert the CAL_LEN_ERR status bit on failure.

51.4.7.2 Successive calibration pulse check

The successive calibration pulse check checks if successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used. Specifically, the higher the frequency, the closer the check will be to 1.5625%. So if protocol clock is T_{HF_CLK} and channel tick period is T_{TX_CLK} then the check will be of 1.5625% ± $\{100 \times (4 \times T_{HF\_CLK}) / (42 \times T_{TX\_CLK})\}$. So it won't allow variance in clock of a channel by 1.5625% - $\{100 \times (4 \times T_{HF\_CLK}) / (42 \times T_{TX\_CLK})\}$. This check will assert the CAL_DIAG_ERR status bit on failure.

When Option 2 of successive calibration pulse check is enabled, the CAL_RESYNC bit will be asserted on every third successive error detected by this check.

51.4.7.3 Not the expected number of edges check

The number of negative edges between two successive calibration pulses is counted and compared with expected number of edges in case of preferred option of successive calibration pulse check. The expected number of nibbles is the sum of following:

- Status and Communication nibble (1 No.)
- Data nibbles (Programmable as per user software)
- CRC nibble (1 No.)
- Pause Pulse, if configured (1 No.)

This check ("Not the expected number of edges check") will assert the NUM_EDGES_ERR status bit on failure. The check does not run when Option 2 (low latency) of successive calibration pulse check is enabled. This check is not masked after any other error is detected so it could be redundant if it is flagged along with CAL_DIAG_ERR and CAL_LEN_ERR. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

51.4.7.4 Nibble value check

As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 nibble values(11.5 ticks to 27.5 ticks):

- Status and Communication nibble
- all Data nibbles
- CRC nibble

The nibble value check will assert the NIB_VAL_ERR status bit on failure. This error can be flagged along with NUM_EDGES_ERR in which case the former can be ignored as the number of pulses in the message is incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. In case, errors are accumulated, the above cannot be deduced.

51.4.7.5 CRC check

The CRC check will compute both 4-bit and 6-bit CRC as requested by State Machine control logic. Two 4-bit CRCs and one 6-bit CRC are supported to compute the CRC on Fast/Slow Serial Message (4-bit CRC) and Enhanced Serial Message (6-bit CRC) respectively. The SENT Receiver Module allows the user to calculate the 4-bit CRC using either the legacy CRC implementation method or the new recommended CRC implementation via a control bit in the channel's corresponding control register. Please refer to SAE SENT Specification for details on CRC algorithm.

Asserting the CHn_CONFIG_REG[11] bit, causes the Status and Communication nibble to be included in the 4-bit CRC calculation for Fast Serial Messages. De-asserting this bit enables normal operation as described above.

This check will assert the FMSG_CRC_ERR or SMSG_CRC_ERR status bit on failure, depending on whether the check is run for Fast Message or Slow Message.

The FMSG_CRC_ERR can be flagged along with the NUM_EDGES_ERR, in which case the CRC error can be ignored as the number of pulses in the message was incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

51.4.7.6 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per spec). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

When the Single Edge Nibble Transmission (SENT) Receiver (SRX) is configured to receive a pause pulse (Channel ‘n' Configuration Register - CHn_CONFIG[PAUSE_EN] = 1) and the length of the pause pulse from the sensor is in range of the valid calibration pulse length, the NUM_EDGES error can get asserted spuriously (Channel ‘n' Status Register - CHn_STATUS(NUM_EDGES_ERR] = 1) when there is any diagnostic error (other than number of expected edges error) or overflow in the incoming messages from the sensor.

Software can distinguish a spurious NUM_EDGES_ERR error from a real one by monitoring other error bits. The following tables will help distinguish between a false and real assertion of NUM_EDGES_ERR error and other errors. Software should handle the first error detected as per application needs and other bits can be evaluated based on these tables. Table 1 contains information due to erratum behavior and Table 2 below contains clarification of normal NUM_EDGES_ERR behavior.

Table 51-5. Behavior of NUM_EDGES_ERR

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR error
FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR errors
CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and	Ignore NUM_EDGES_ERR error

Table 51-5. Behavior of NUM_EDGES_ERR

		start of bad message (which has CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	

Table 51-6. Expected behavior, clarification of NUM_EDGES_ERR cases

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles when actually there is a pause pulse present. This generates NIB_VAL_ERR.	Ignore the NIB_VAL_ERR
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	Ignore NIB_VAL_ERR and PP_DIAG_ERR

51.4.8 Time stamp logic

A single counter maintains the time stamp for all channels which can be used by the software to determine the relative time of arrival of the messages for each channel. The time stamp counter is a 32-bit counter.

The clock for this counter is derived from the high frequency receiver clock using a Time Stamp Prescaler Counter that divides the high frequency receiver clock to generate the clock of required resolution. The user must program the time stamp prescaler value (in Global Control Register (SRX_GBL_CTRL)) in order to generate the time stamp clock. The period of the time stamp clock must be less than that of the fastest receiver channel being used, or it can be set at some fixed value determined by application, such as 1 µs. For example, if the high frequency receiver clock is 60 MHz, the required prescaler value is 59 to generate a time stamp clock of 1 µs resolution.

Note

> The rollover of time stamp values should be determined in user software, by comparing the time stamp values in previous and current messages that are read out. User software should adjust the time stamp values in these messages accordingly, after reading them.

The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer. However, if the current message is found to be in error, the message buffer is discarded. Thus, the time stamp will be available to user software for only those messages which do not have any error.

The time stamp for a serial message is sampled at the end of the message i.e. when the serial message is completely received and when the message is found to have no errors. the time stamp is stored in the message read buffer to be read by the user software. Again, for messages with error the message and time stamp are discarded.

51.4.8.1 Limitation

Since the same counter is used for all channels, it may happen that more than 1 message is received within the same time stamp across different channels. Hence the software would not be able to determine the sequence of arrival of messages which have the same time stamp value.

51.4.9 Bus Idle Diagnostic

This diagnostic checks the connected sensors for inactivity. SENT module can indicate if the idle period on a particular channel has crossed the period defined by the programmed value in Bus Idle Count (in Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)). Status bit (bus_idle in Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)) will be set when idle period crosses the allowed value. CPU should write 1 to this clear this bit once it is read.

51.5 Clocks and resets

The SENT Receiver works on two clocks. First is the System Bus Clock that is used to program the registers and read messages via DMA or Interrupt from the module's Register Interface. Second is the High Frequency Receiver Clock or Protocol Clock that is used for accurate message receiving operations. There is a single asynchronous reset which is the system asynchronous reset and one internal reset for each of the channel logic.

51.5.1 Clocking strategy

The following figure shows the clock domain in which each module is functioning.

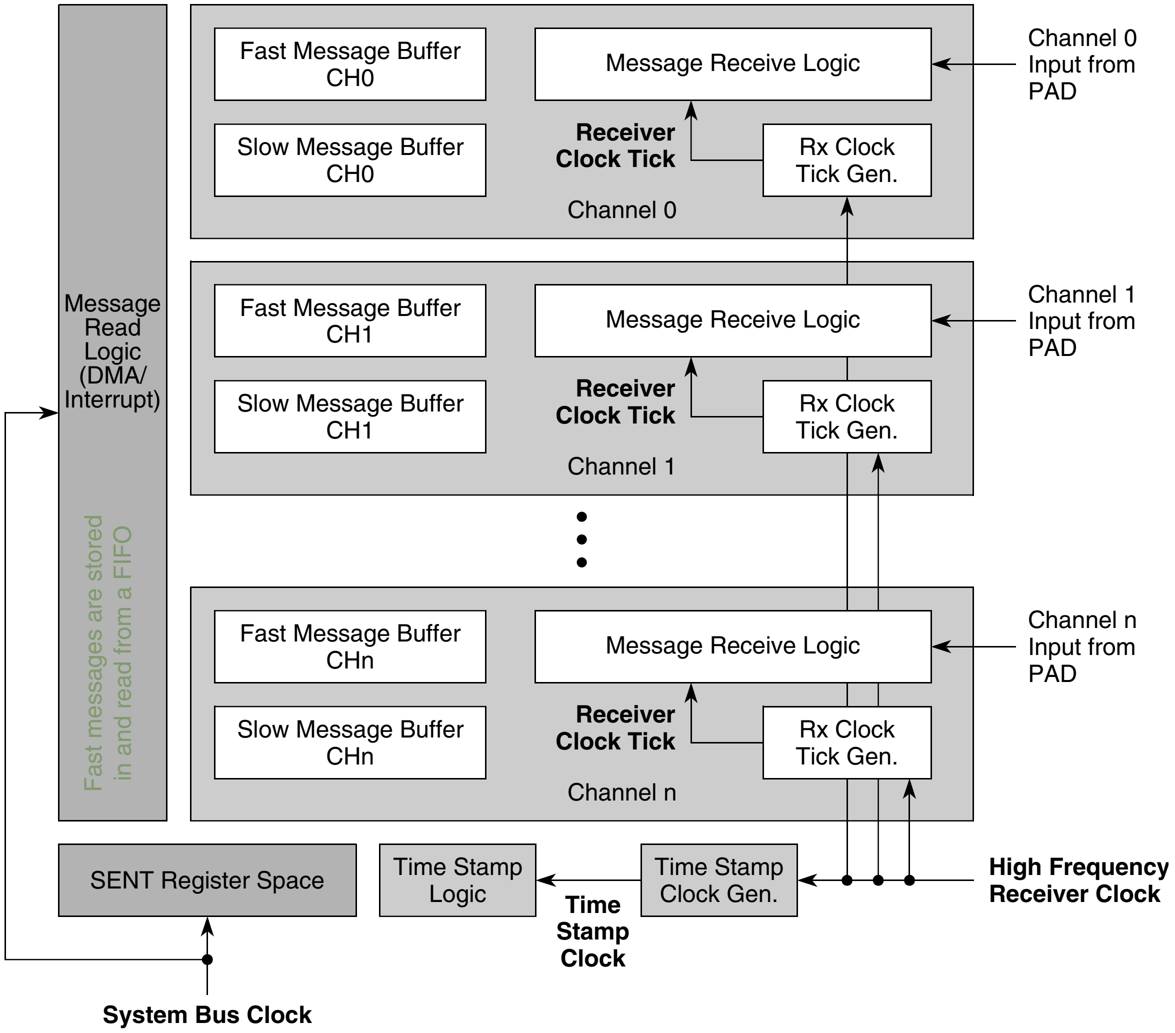


Figure 51-6. SENT Receiver module clock domains

51.5.2 System bus clock requirements

Since all the received messages are read on the System Bus Clock it is necessary that both the interrupt service routine and the DMA controller must be able to access all channels with messages ready to be read within a time to prevent overflow in any channel. The minimum time available to the CPU or DMA is the case when the smallest Fast Message is received back to back on all channels while working on the smallest receive clock tick period.

Now, the smallest Fast Message size with valid checksum as per SAE Specifications is 154 ticks. At 3 µs receive clock tick with a -20% variation in the transmitter clock tick, the effective receiver clock tick is 2.4 µs (after compensation). This gives us the time in which the minimum size Fast Message is received in any channel as 154 × 2.4 = 369.6 µs. Now since messages are arriving in parallel, 369.6 µs is the time available for CPU or DMA to access all channels to prevent an overflow in the channel that received the very first message.

Thus, if assume that 'N_{CHNL}' channels are being used and all channels have messages ready to be read together from either DMA or CPU then the time available to read a message in one channel will be 369.6/N_{CHNL} µs or ~370/N_{CHNL} µs. Since the CPU or the DMA needs to make three 32-bit accesses in order to read a complete message and taking the time required for one 32-bit read access as 'T_{RD}' (assuming it to be same for DMA and CPU); we can determine the minimum frequency for the System Bus Clock as:

For CPU,

$$3 \times T_{\mathrm{RD}} + T_{\mathrm{ISR}} + T_{\mathrm{WAKUP}} < 370 / N_{\mathrm{CHNL}}$$

where T_{ISR} is the overhead in initiating an ISR and T_{WAKUP} is the time to wakeup from low power mode

For DMA,

$$3 \times T_{\mathrm{RD}} + T_{\mathrm{WAKUP}} < 370 / N_{\mathrm{CHNL}}$$

Replacing $T_{RD} = N_{RD} \times T_{SYS\_CLK}$ and $T_{ISR} = N_{ISR} \times T_{SYS\_CLK}$, where N_{RD} is the number of System Bus Clock cycles in one 32-bit read access, N_{ISR} is the number of System Bus Clock Cycles in initiating an interrupt service routine and T_{SYS_CLK} is the period of the same clock, we get

For CPU,

$$(3 \times N_{\mathrm{RD}} + N_{\mathrm{ISR}}) \times T_{\mathrm{SYS\_CLK}} + T_{\mathrm{WAKUP}} < 370 / N_{\mathrm{CHNL}}$$

(all units is µs)

For DMA,

$$3 \times N_{RD} \times T_{SYS\_CLK} + T_{WAKUP} < 370 / N_{CHNL}$$

(all units is µs)

51.5.3 High frequency receiver clock (protocol clock) requirements

For accurate receive operations, the protocol is required to satisfy a minimum frequency defined as per formula: Guard Band > 2.5 * T_{HF_CLK}

where T_{HF_CLK} is the High Frequency Receiver Clock or Protocol Clock in ns and Guard Band is guard band in ns as specified in SAE specification, at particular transmitter uTick.

Note

> T_{HF_CLK} in the above equation should be chosen based on the fastest sensor clock (with minimum guard band as specified in SAE spec) among all channels in the device.

Chapter 52
LINFlexD

52.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, and 2.1. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/UART operating mode.

52.1.1 Glossary and acronyms

Table 52-1. Glossary and acronyms

Term	Description
LIN	Local interconnect network
UART	Universal asynchronous transmitter receiver
ID	LIN identifier field
DMA	Direct memory access

Table continues on the next page...

Table 52-1. Glossary and acronyms (continued)

Term	Description
TCD	Transfer control descriptor
IPS	Peripheral Bus Interface
IRQ	Interrupt request
FSM	Finite state machine
WS	Wait state
SW	Software
CPU	Central processing unit
SoC	System on a chip

52.1.2 References

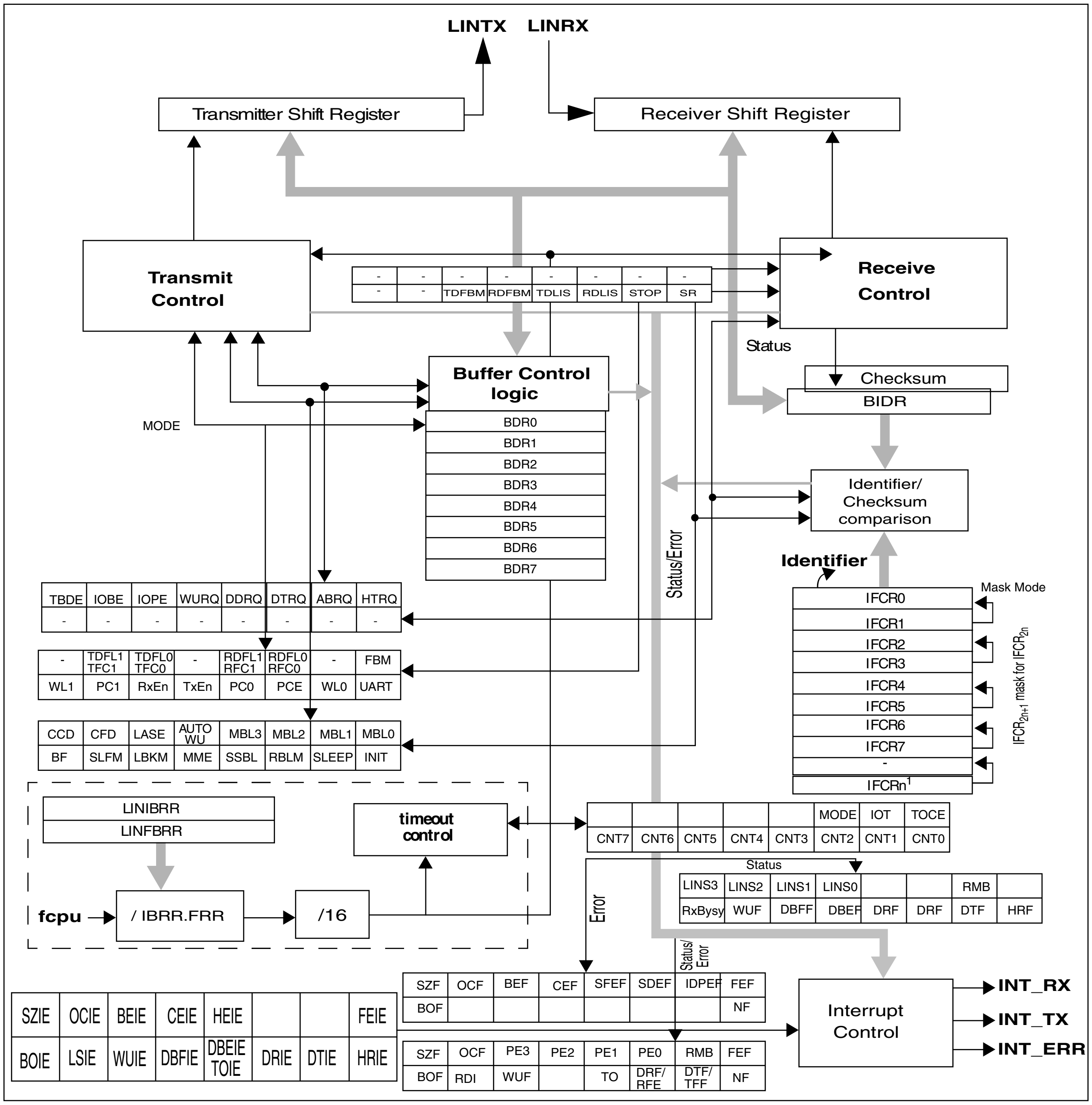


Figure 52-1. Block diagram

Notes:

1: The value of n depends on the no_of_filters. Refer to the chip configuration details to see the number of filters used in this device.

52.2 Main features

The LINFlexD controller can operate in several modes, each of which has a distinct set of features that are described in the following sections. In addition, the LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
    - Initialization
    - Normal
    - Sleep
- Test Mode:Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

52.2.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, and 2.1
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection

- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
    - Autonomous header handling
    - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation

52.2.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers - see Baud rate generation
- Separate clock for baud rate calculation
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management

52.3 Functional description

52.3.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

52.3.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

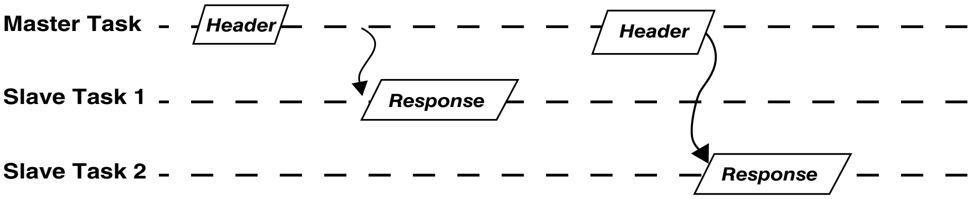


Figure 52-2. Frames

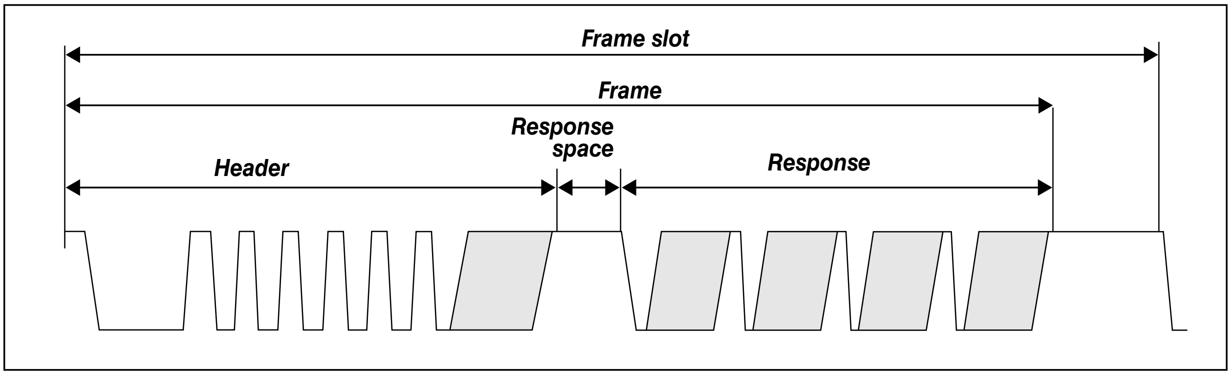


Figure 52-3. Structure of LIN frame

52.3.1.2 Data field

Each byte is transmitted as shown in Figure 52-4. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

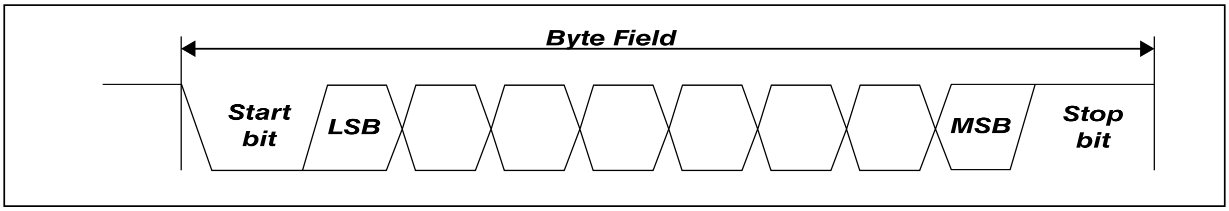


Figure 52-4. Structure of byte field

52.3.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in Figure 52-5. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

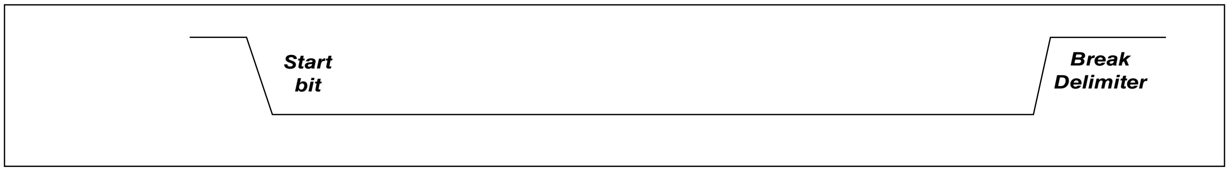


Figure 52-5. Break field

52.3.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

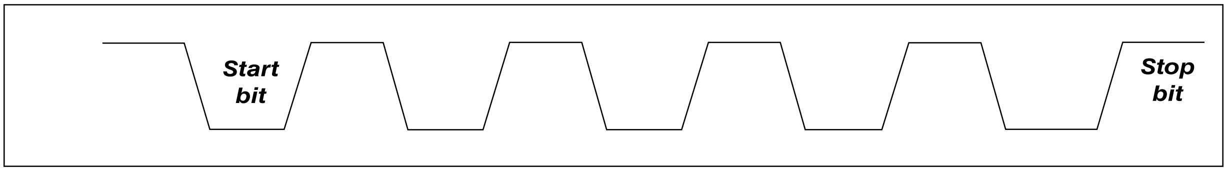


Figure 52-6. Sync byte field

52.3.1.5 Identifer

The Identifier field consists of two sub-fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

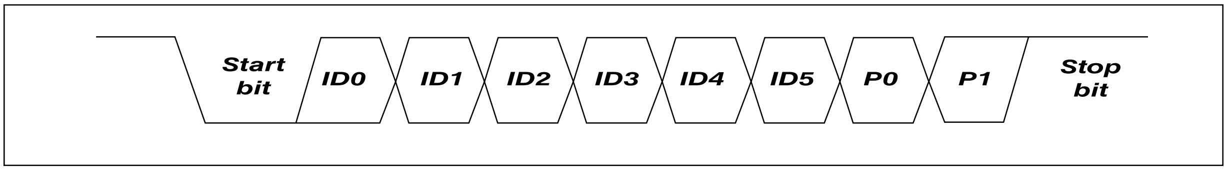


Figure 52-7. Identifier field

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

52.3.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and is used for communication with LIN 2.0 slaves.

52.3.2 LINFlexD features

52.3.2.1 Operating modes

The LINFlexD has three operating modes: **Initialization, Normal** and **Sleep** modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINCR1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller software must:

1. Set up the baud rate registers
2. Reset UART bit
3. Select the mode (master or slave)
4. Configure checksum control bits
5. Initialize the identifier list (Slave mode)

Normal mode (NM)

Once software has completed initialization of the LINFlexD controller, it can enter the Normal mode by clearing the INIT bit.

Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINCR1. In this mode the LINFlexD clock is stopped. LINFlexD can be awakened from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 μs on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to Wakeup management.

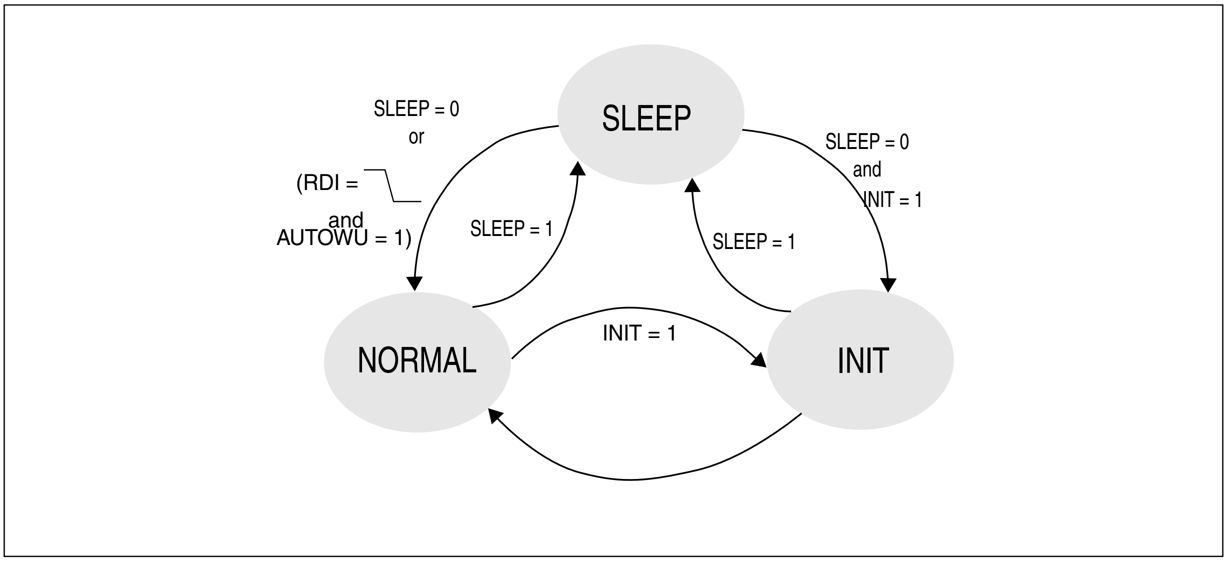


Figure 52-8. Operating modes

52.3.2.2 Test mode

Loop Back mode

This mode is entered by setting the LBKM bit in LINCR1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is an internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.

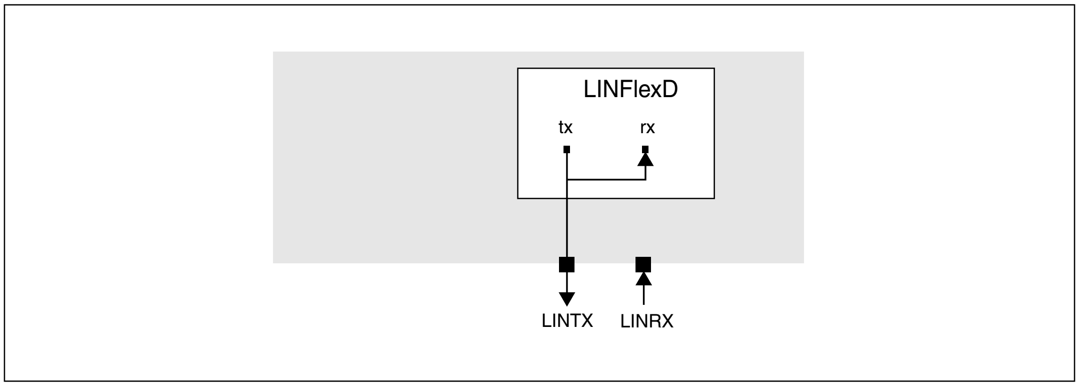


Figure 52-9. LINFlexD in Loop Back Mode

52.3.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits untill the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDRL and BDRM. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling).

It is possible to handle frames with a response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, LINSR[DBEF] is set once the first eight bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset. Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to Error handling).

It is possible to handle frames with a Response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, the LINSR (DBFF) bit is set once the first eight bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

52.3.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. Read the received ID in the BIDR register
2. Fill the BDR[0:x] register
3. Program the CCS and DIR bits in the BIDR register
4. Specify the data field length DFL[5:0] bits in the BIDR register
5. Trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit, and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7].

The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software then you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BIDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

52.3.2.5 Errors

52.3.2.5.1 Header error

A header error is an error during the header reception. The error types are:

1. Sync Del error (SDEF)
2. Sync field error (SFEF)
3. Identifier Parity error (IDPEF)

Sync Del error (SDEF)

The delimiter should be 1 for at least one bit time; otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if *HEIE* bit of LINEIER is set.

Sync field error (SFEF)

The SFEF error condition is monitored differently depending on whether autosynchronization (LASE) is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the SFEF flag indicates:

- The deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators or
- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator).

This check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

If SFEF field is asserted it means that:

$D > 14.0625\%$

If there is no error, it means that:

$D < 14.84375\%$

If $14.0625\% < D < 14.84375\%$, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.

Overflow during Sync Field Measurement

This check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = '0')

In this case the Sync character is received as a normal character. If the received character is 0x55 then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

Identifier Parity error(IDPEF)

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

52.3.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CPU cycles. Bit error is not checked during break field transmission.

1-bit time at 20 Kbit/s = 50 µs

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) - (6 LIN_CLK cycles) = 49.925 µs

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note

If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.

52.3.2.5.3 Framing error

This error is flagged when a dominant state is sampled on stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if FEIE bit is set in LINEIER.

The byte that caused framing error is also shifted to the buffer but DRF or DBFF is never set in this case.

52.3.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINEIER.

52.3.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of slave, if buffer is not released (RMB is not reset) before reception of next Identifier and if RBLM is set, then the ID along with the data is discarded.

52.3.2.5.6 Timeout error

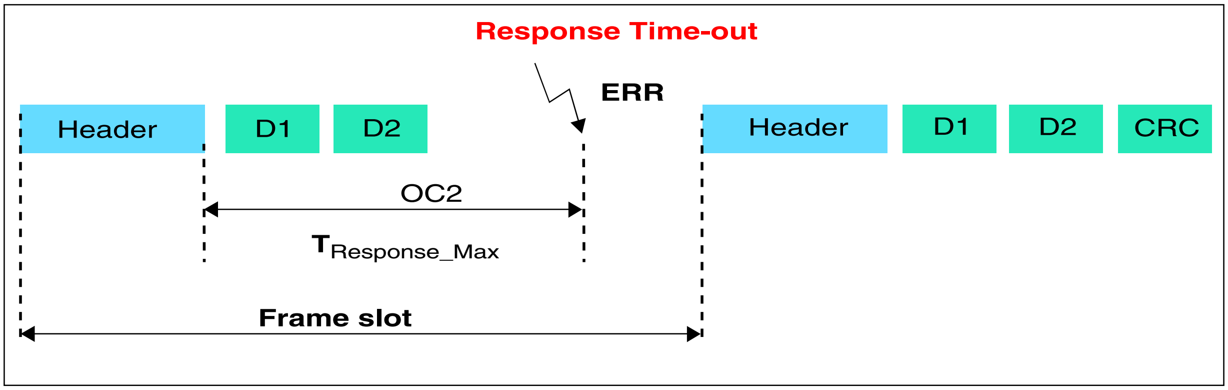


Figure 52-10. Incomplete response (for example, missing checksum)

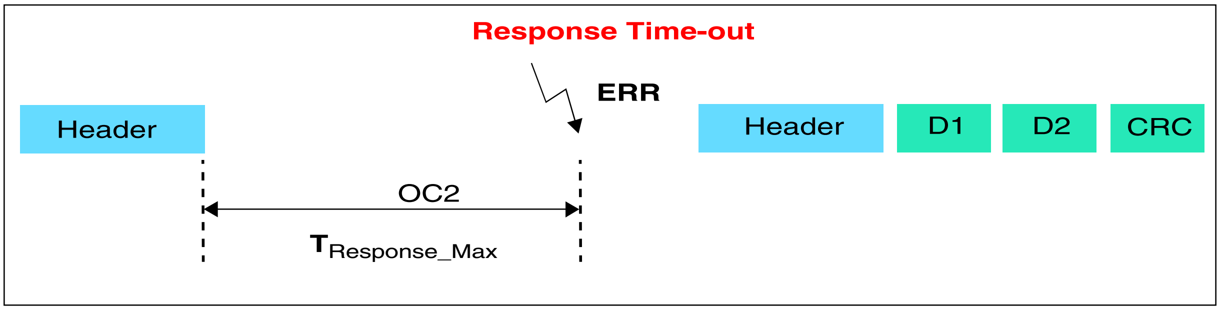


Figure 52-11. No response

Response timeout mechanism

- Master mode
  - OC2 is loaded with Nominal_time_out + LINFlexD_LINTCSR[CNT] at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time).

    This loading takes place at the end of id field (and not at the beginning of data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further OC2 update is required.
  - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the OC2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the OC2 value (no change compared to earlier implementation).
- Slave mode

- **Case 1**: The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, OC2 can be loaded with Nominal_time_out + LINFlexD_LINTCSR[CNT] at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time))
- **Case 2**: The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the implementation is the following:
    - At the end of the ID, OC2 is loaded with 36 (maximum possible response space) + LINFlexD_LINTCSR[CNT].
    - At the end of the first_data_byte it is reloaded again according to DFL
    - Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

NOTE

The value of nominal_time_out (1.4 × ((DFL +2) × 10 bit time)) shown is as per the reset value of LINTOCR[RTO] and LINCR1[CFD]. It changes with change in value of LINTOCR[RTO] and LINCR1[CFD]. Here, 1.4 corresponds to (LINTOCR[RTO])/10 and 2 corresponds to (2- LINCR1[CFD])

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
    - no error on the bus and timing is correct (nominal header length),
    - an error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error).

  Therefore there is no meaning of header timeout in master mode, so it is disabled.
- Slave mode
    - header_nominal = 13 + 2 + 10 + 10 = 35 Tbit
    - header_max = 1.4 × Header_nominal = 49 Tbit

- taking into account a possible 14% clock deviation, header_max seen by LINFlexD is 49 × 1.14 = 56 Tbit
- If the counter starts after 11 Tbit (break duration), the HTO value is 56 –11 = 45 Tbit.

The reset value of HTO is 45, and this register can only be programmed in slave mode. Counter restarts after break duration and OC1 gets loaded with the value of HTO.

As response space is not included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be reset and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

52.3.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8th, 9th and 10th samples. If any one of these three samples has a value different from the other two, this error is flagged.

This error is flagged when there is noise detected in the start bit (see LIN Error Status Register (LINFlexD_LINESR).

52.3.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2n and 2n+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2n or IFCR2n+1 (if the corresponding FACT bit is set).

Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2n is the filter and 2n+1 acts as a mask for it. The FACT bit for filter 2n+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

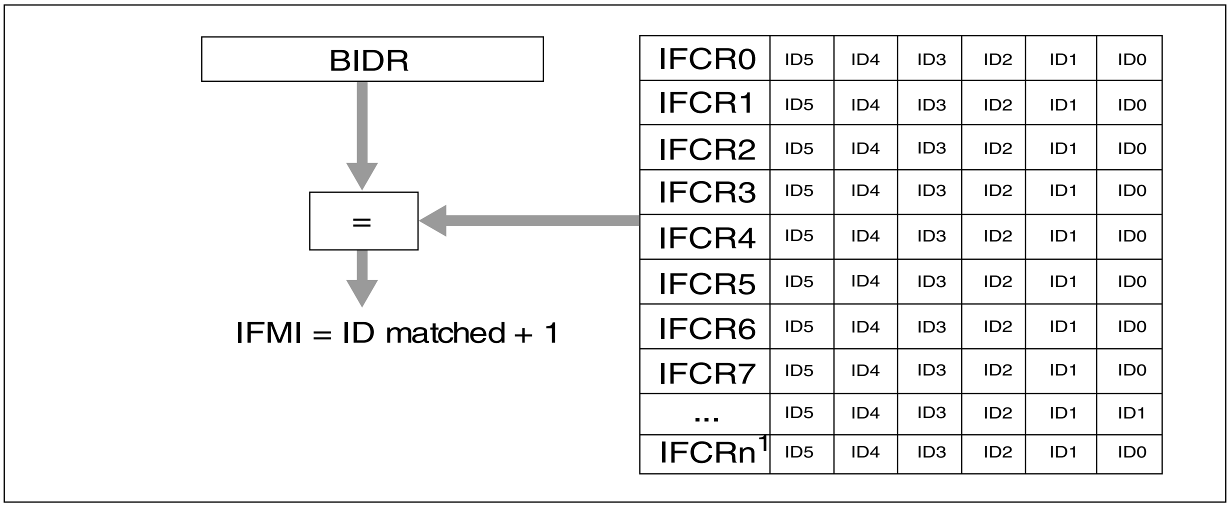
If there is a match of identifier with the mth filter in any mode, then m+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only untill the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

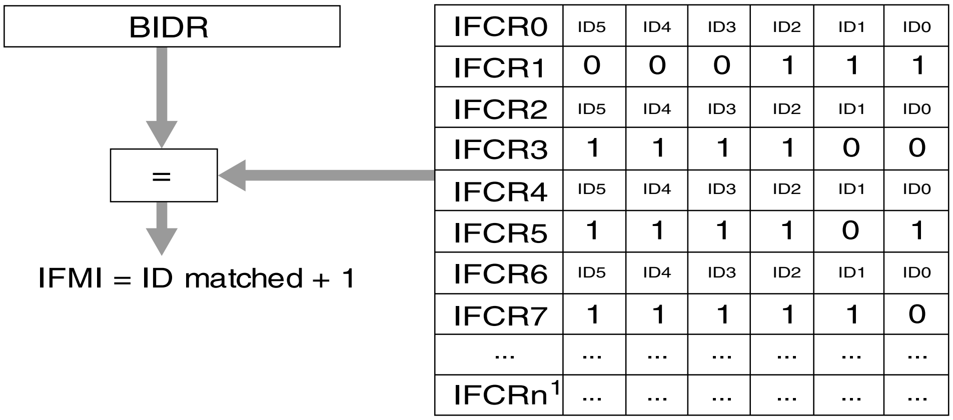
Note

If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In mask mode if two filters match with the identifier then the filter having the lower number prevails.



IFCR0	ID5	ID4	ID3	ID2	ID1	ID0
IFCR1	ID5	ID4	ID3	ID2	ID1	ID0
IFCR2	ID5	ID4	ID3	ID2	ID1	ID0
IFCR3	ID5	ID4	ID3	ID2	ID1	ID0
IFCR4	ID5	ID4	ID3	ID2	ID1	ID0
IFCR5	ID5	ID4	ID3	ID2	ID1	ID0
IFCR6	ID5	ID4	ID3	ID2	ID1	ID0
IFCR7	ID5	ID4	ID3	ID2	ID1	ID0
...	ID5	ID4	ID3	ID2	ID1	ID1
IFCRn[1]	ID5	ID4	ID3	ID2	ID1	ID0

Figure 52-12. Identifier List mode



IFCR0	ID5	ID4	ID3	ID2	ID1	ID0
IFCR1	0	0	0	1	1	1
IFCR2	ID5	ID4	ID3	ID2	ID1	ID0
IFCR3	1	1	1	1	0	0
IFCR4	ID5	ID4	ID3	ID2	ID1	ID0
IFCR5	1	1	1	1	0	1
IFCR6	ID5	ID4	ID3	ID2	ID1	ID0
IFCR7	1	1	1	1	1	0
...	...	...	...	...	...	...
IFCRn[1]	...	...	...	...	...	...

$IFCR_{2n+1}$ = Mask for IFCR ^{2n}
1 = must match bits
0 = don't care bits

Figure 52-13. Identifier Mask mode

52.3.2.7 Start detection and break delimiter detection in receiver

There is a 10-bit-shift register sample register in the receiver that shifts signal data to the next least significant bit on each incoming sample.

Table 52-2. Start Detection and Delimiter Detection in receiver

9	8	7	6	5	4	3	2	1	0	Sample_reg
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x	0	x	0	x	0	start detect
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	delimiter detect
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

52.3.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to Table 52-2 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to ‘0’, then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to ‘0’ then Noise flag is set.

52.3.2.7.2 Break delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to Table 52-2 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to ‘1’, then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

52.3.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate Register and the LIN Fraction baud rate register. The Baud Rate Registers can be programmed only during Initialization mode.

Baud rate is calculated with the following formula for both receiver and transmitter.

Tx = Rx = LIN_CLK / (16 × LDIV)

Where LIN_CLK is the frequency of the baud clock.

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBRR.

52.3.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescalar value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When auto synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

52.3.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 µs to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 µs) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

In LINFlexD, wakeup can be detected in two ways.

1. AUTOWU = 1 On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated if WUPIE is set. LINFlexD is now in normal mode and ready to receive frames.
2. AUTOWU = 0 On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE bit is set). It is then up to the software to clear the SLEEP bit.

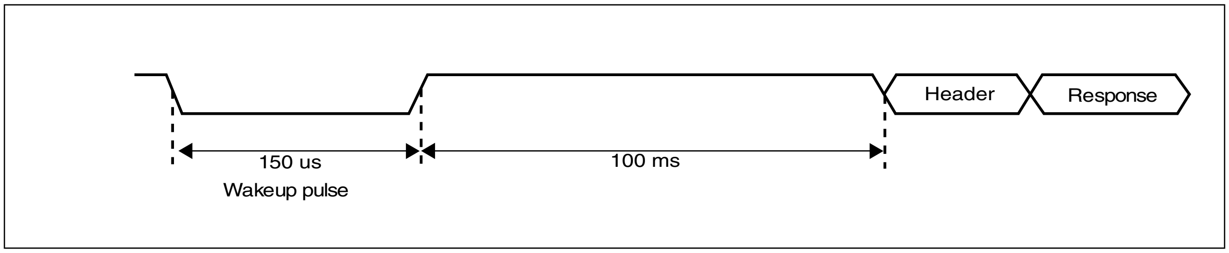


Figure 52-14. Wakeup sequence

52.3.3 Timer

There is an 8-bit counter which has different behavior in different modes as described below:

- **In Output Compare Mode (LINTCSR[MODE]= 1):**

  This counter is running even in Sleep and Initialization mode. The counter value which when matches the two software configurable output compare registers (LINOCR[OC1] or LINOCR[OC2]), generate output compare interrupt (LINESR[OCF]) provided TOCE bit in LINTCSR is set. Once an interrupt occurs subsequent interrupt is generated only when the application has reset the stuck FSM state.
- **In LIN Mode:**

The software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

52.3.4 UART mode

Main features in the UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames, 13-bit frames, 16-bit frames, 17-bit frames
- Even/Odd/0/1 Parity

52.3.4.1 8-bit data frames

The eighth bit can be a data or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

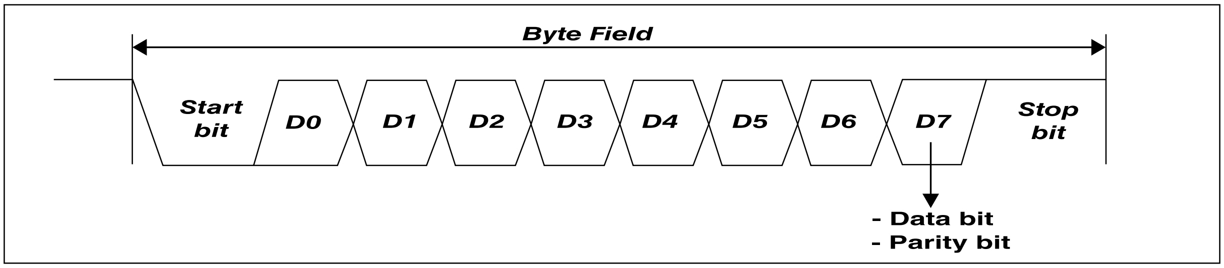


Figure 52-15. UART mode 8-bit data frame

52.3.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

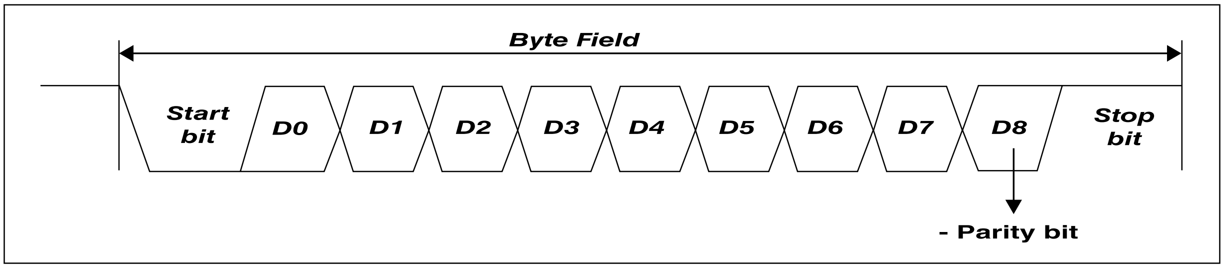


Figure 52-16. UART mode 9-bit data frame

52.3.4.3 16-bit data frames

The sixteenth bit can be a data or a parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

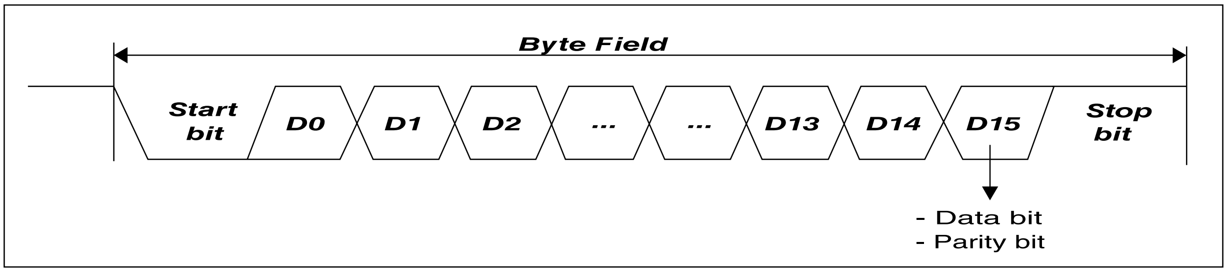


Figure 52-17. UART mode 16-bit data frame

52.3.4.4 17-bit frames

The seventeenth bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

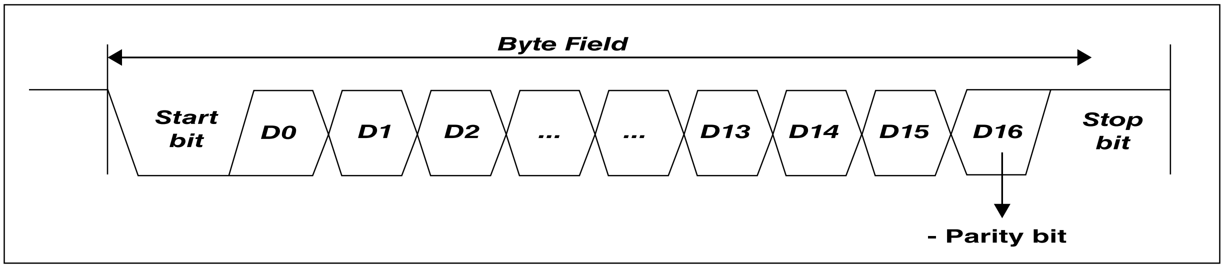


Figure 52-18. UART mode 17-bit data frame

52.3.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception only in FIFO mode.

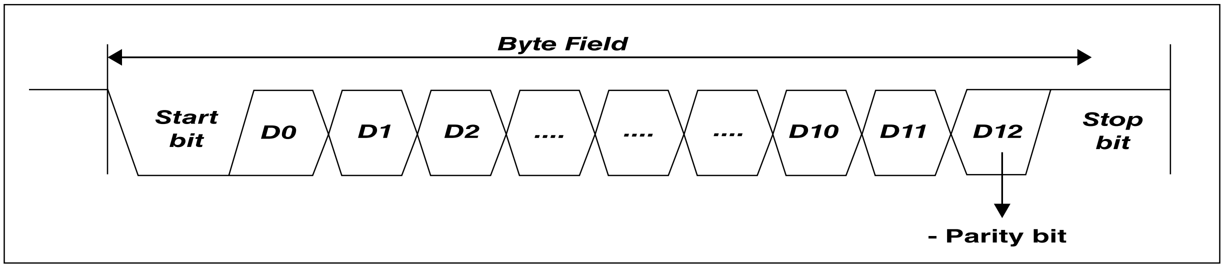


Figure 52-19. UART mode 13-bit data frame

52.3.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in the following figure:

Tx0	BDR0
Tx1	BDR1
Tx2	BDR2
Tx3	BDR3
Rx0	BDR4
Rx1	BDR5
Rx2	BDR6
Rx3	BDR7

Figure 52-20. Structure of 8-byte buffer

For 16-bit frames, the lower eight bits are written in BDR0 and upper eight bits in BDR1.

The same applies for reception.

52.3.4.7 UART transmitter

UART transmitter

In order to start transmission in the UART mode, UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the TxEn bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

NOTE

If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Table 52-3. BDRL access in UART mode

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR[WL])	IPS operation result
Write Byte0	BDRL	FIFO	Byte	OK
Write Byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write Half-word0-1	BDRL	FIFO	Byte	IPS transfer error
Write Word	BDRL	FIFO	Byte	IPS transfer error
Write Byte0-1-2-3	BDRL	FIFO	Half-word	IPS transfer error
Write Half-word0	BDRL	FIFO	Half-word	OK
Write Half-word1	BDRL	FIFO	Half-word	IPS transfer error
Write Word	BDRL	FIFO	Half-word	IPS transfer error
Read Byte0-1-2-3	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Half-word0-1	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Word	BDRL	FIFO	Byte/Half-word	IPS transfer error
Write Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Write Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK
Write Word	BDRL	BUFFER	Byte/Half-word	OK
Read Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Read Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK
Read Word	BDRL	BUFFER	Byte/Half-word	OK

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

52.3.4.8 UART receiver

UART receiver

The reception of a data byte is started as soon as the user exits initialization mode, sets the RxEn bit, and detects start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-half-words (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. RxEn bit needs to be set only to start the reception.The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 52-4. BDRM access in UART mode

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read Byte4	BDRM	FIFO	Byte	OK
Read Byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read Half-word2-3	BDRM	FIFO	Byte	IPS transfer error
Read Word	BDRM	FIFO	Byte	IPS transfer error
Read Byte4-5-6-7	BDRM	FIFO	Half-word	IPS transfer error
Read Half-word2	BDRM	FIFO	Half-word	OK
Read Half-word3	BDRM	FIFO	Half-word	IPS transfer error
Read Word	BDRM	FIFO	Half-word	IPS transfer error
Write Byte4-5-6-7	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Word	BDRM	FIFO	Byte/Half-word	IPS transfer error
Read Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	OK
Read Half-word2-3	BDRM	BUFFER	Byte/Half-word	OK
Read Word	BDRM	BUFFER	Byte/Half-word	OK
Write Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Word	BDRM	BUFFER	Byte/Half-word	IPS transfer error

Note

Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and data bits of the registers BDRL/BDRM.

Note

- If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.
- If RDFL is programmed for a certain value but that number of bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to Idle state.
- If a STOP request arrives in the middle of one reception, it is only acknowledged after all the programmed number of data bytes have received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout. When the state machine moves to Idle state, then only a stop request is served.
- If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if FEIE bit in LINIER is set. As there is only one register bit for framing error, this interrupt will be helpful in identifying which byte has framing error.
- If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is depending on RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.
- When WLS bit = 1, BDRM access depends on WL bit setting.

- When WLS bit = 1, WL = 0 or 1 should not be used, since this will lead to incorrect reception of data.
- When IPG_STOP is requested, for UART in FIFO mode, SW has to set the INIT bit in order to send the receiver to idle. This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

52.3.5 DMA interface

52.3.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channel where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channel where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

52.3.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in Table 52-5 and Table 52-6.

Table 52-5. LINFlexD control/status fields description

Register	Field	Level	Description
LINCR2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)

Table continues on the next page...

Table 52-5. LINFlexD control/status fields description (continued)

Register	Field	Level	Description
BIDR	DFL [5:0]	—	Data field length: • 1 to 8 bytes: normal frames • 1 to 64 bytes: extended frames
	DIR	—	Direction of the data field: • 0 => reception • 1 => transmission
	CCS	—	Checksum type: • 0 enhanced • 1 classic
	ID[5:0]	—	Identifier.
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA. • 0 buffer data is free • 1 buffer data is ready
	DBFF	High	Data buffer full (8 bytes received) and DFL > 7. Used for the extended frames.
	DBEF	High	Data buffer empty (8 bytes have been transmitted) and DFL > 7. Used for the extended frames.
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received. Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode).
UARTCR	FBM	—	FIFO/Buffer mode • 0 Buffer mode • 1 FIFO mode (mandatory in DMA mode)
UARTSR	RFE	—	Rx FIFO empty • 0 Rx FIFO not empty • 1 Rx FIFO empty
	TFF	—	Tx FIFO full • 0 Tx FIFO not full • 1 Tx FIFO full

DMATXE[2**TX_CH_NUM][1]		High	DMA Tx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed.
			In LIN slave mode: # DMA TX channel = IFMI[3:0] -1.
			DMATXE[x] = 0b => TX channel x disabled
			DMATXE[x] = 1b => TX channel x enabled

DMARXE[2**RX_CH_NUM][1]		High	DMA Rx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed.

Table 52-5. LINFlexD control/status fields description

Register	Field	Level	Description

			In LIN slave mode: # DMA RX channel = IFMI[3:0] -1.
			DMARXE[x] = 0b => RX channel x disabled
			DMARXE[x] = 1b => RX channel x enabled

1. ** stands for exponentiation.

1. ** stands for exponentiation

Table 52-6. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0]	High	Logical AND between the 2 DMA enable bits (LINFlexD and eDMA).
DMA_TEN	LIN slave Tx	DMATXE[x] x = IFMI — 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] x = IFMI — 1	High	

Control/status fields of the TCD descriptors referred to in this document are described in Table 52-7.

Table 52-7. TCD control fields description

TCD Field	Level	Description
CITER[14:0]	—	Current "major" iteration count
BITER[14:0]	—	Beginning "major" iteration count
NBYTES[31:0]	—	Inner "minor" byte transfer count. Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset applied to the current source address as each source read is completed.
SSIZE[2:0]	—	Source data transfer size 000 => 8-bit 001 => 16-bit 010 => 32-bit 011 => 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address

Table continues on the next page...

Table 52-7. TCD control fields description (continued)

TCD Field	Level	Description
DOFF[15:0]	—	Destination address signed offset applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 => 8-bit 001 => 16-bit 010 => 32-bit 011 => 64-bit
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request.
DONE	High	Channel done (the DMA has completed the outer major loop).
D_REQ	High	Disable request. If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

52.3.5.3 Master node –TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

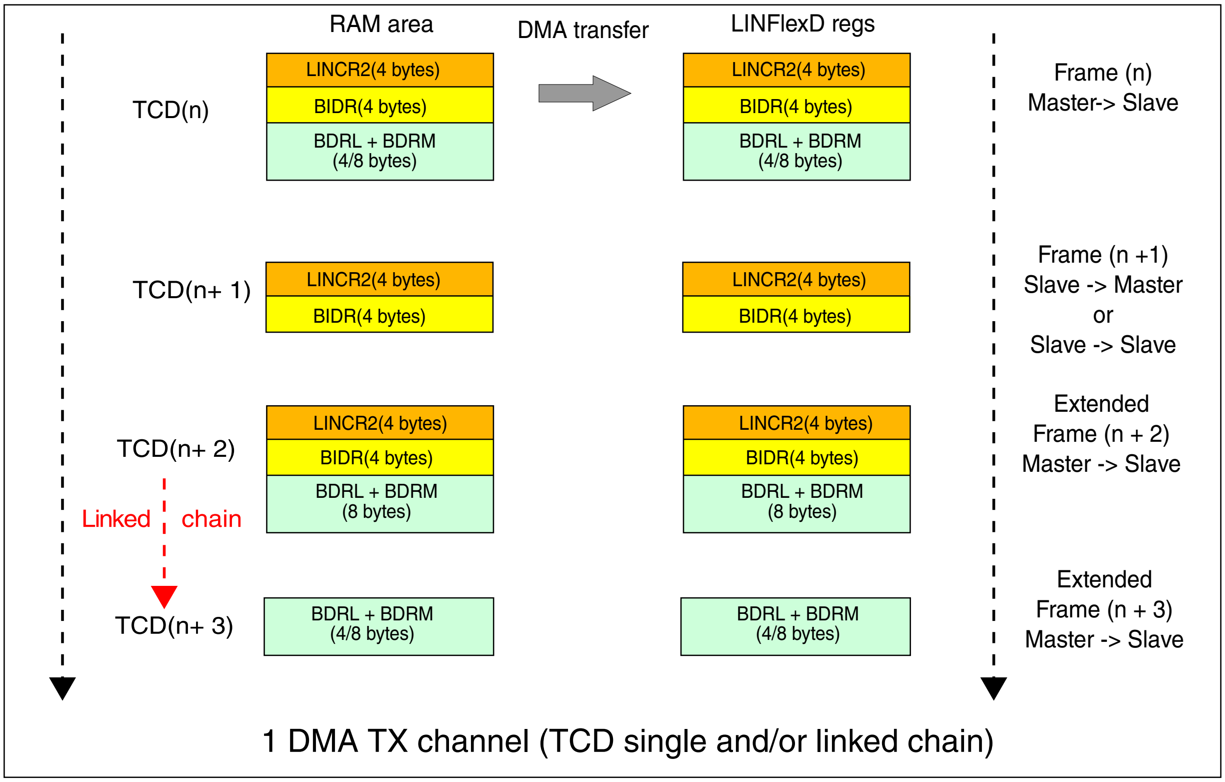


Figure 52-21. Master node –TX memory map

The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

The following table provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 52-8. Master node – Tx mode – Register setting

LIN frame	LINCR2	BIDR

Master to Slave	DDRQ = 1
	DTRQ = 0
	HTRQ = 0
		DFL = payload size
		ID = address
		CCS = checksum
		DIR = 1 (TX)

Slave to Master	DDRQ = 0
	DTRQ = 0
	HTRQ = 0
		DFL = payload size
		ID = address
		CCS = checksum
		DIR = 0 (RX)

Table continues on the next page...

Table 52-8. Master node – Tx mode – Register setting (continued)

LIN frame	LINCR2	BIDR

Slave to Slave	DDRQ = 1
	DTRQ = 0
	HTRQ = 0
		DFL = payload size
		ID = address
		CCS = checksum
		DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in Table 52-9. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 52-9. TCD setting – Master node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4 + 4] + 0/4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. LINCR2 + BIDR + BDRL + BDRM
SADDR[31:0]	-	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	
DADDR[31:0]	-	LINCR2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

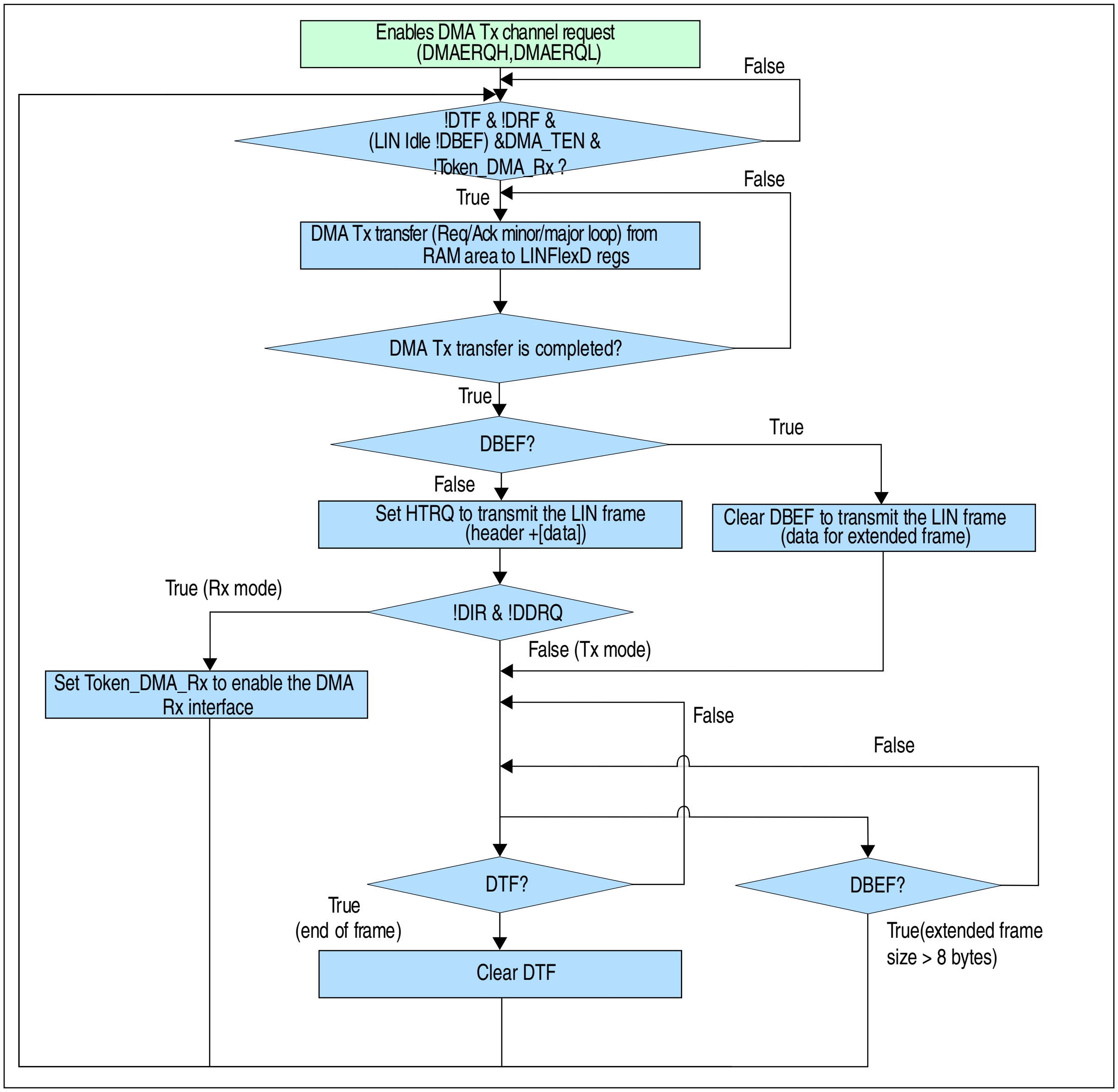

Enables DMA Tx channel request
(DMAERQH,DMAERQL)
False
!DTF & !DRF &
(LIN Idle !DBEF) &DMA_TEN &
!Token_DMA_Rx ?
False
True
DMA Tx transfer (Req/Ack minor/major loop) from
RAM area to LINFlexD regs
DMA Tx transfer is completed?
True
DBEF?
True
False
Set HTRQ to transmit the LIN frame
(header +[data])
Clear DBEF to transmit the LIN frame
(data for extended frame)
True (Rx mode)
!DIR & !DDRQ
False (Tx mode)
Set Token_DMA_Rx to enable the DMA
Rx interface
False
False
DTF?
DBEF?
True
(end of frame)
True(extended frame
size > 8 bytes)
Clear DTF

Figure 52-22. Master node – DMA Tx FSM (concept scheme)

52.3.5.4 Master node - RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame , except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

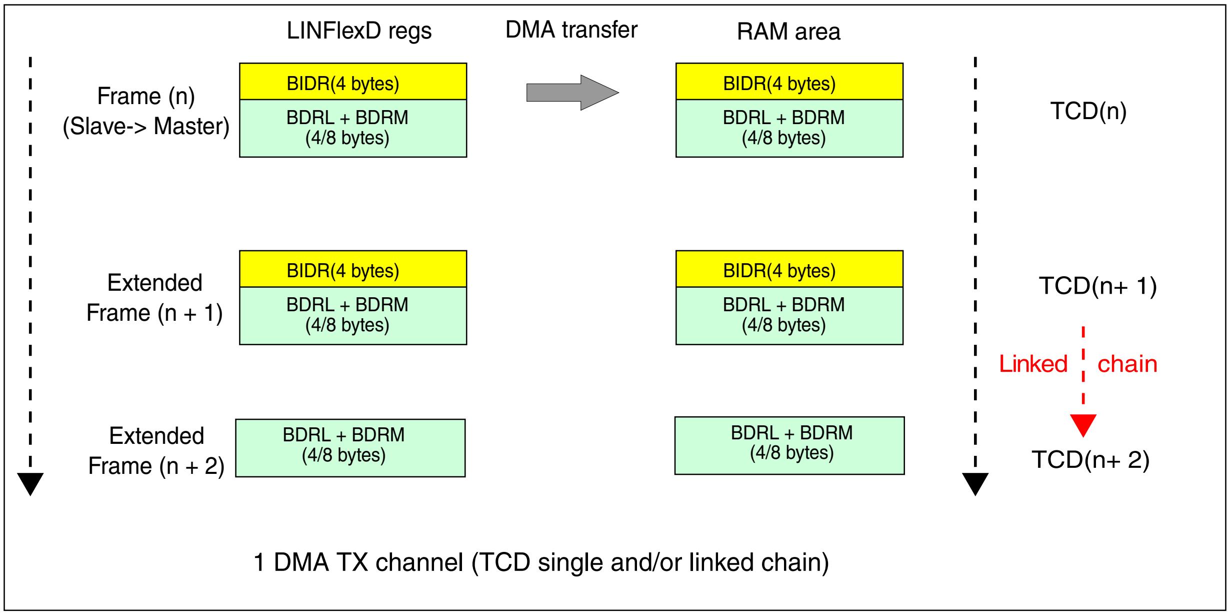


Figure 52-23. Master node – RX memory map

The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in the following figure. DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in the next table. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 52-10. TCD setting – Master node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] +4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer

Table continues on the next page...

Table 52-10. TCD setting – Master node – Rx mode (continued)

TCD Field	Value	Description
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

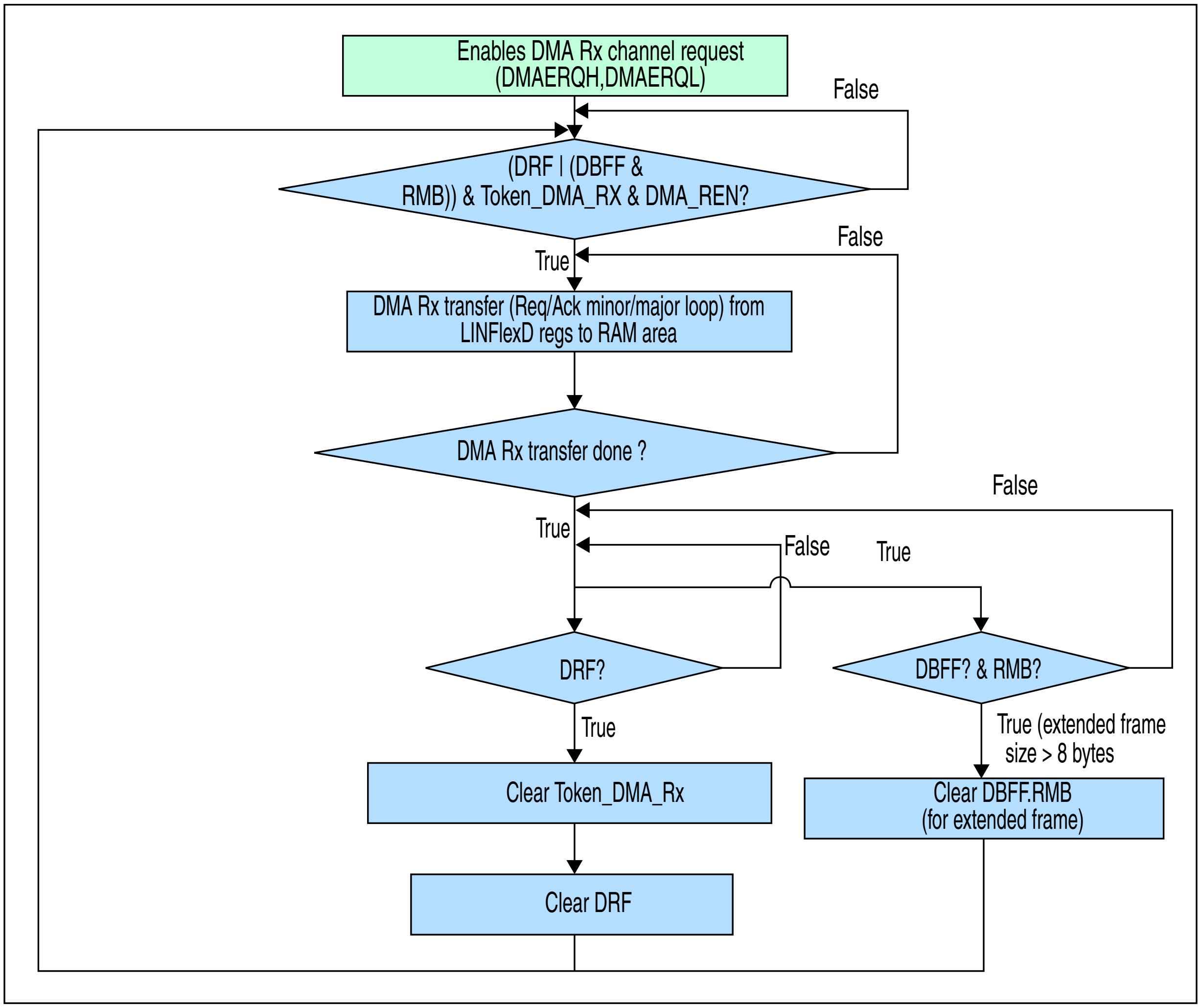


Figure 52-24. Master node – DMA Rx FSM (concept scheme)

52.3.5.5 Slave node – TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

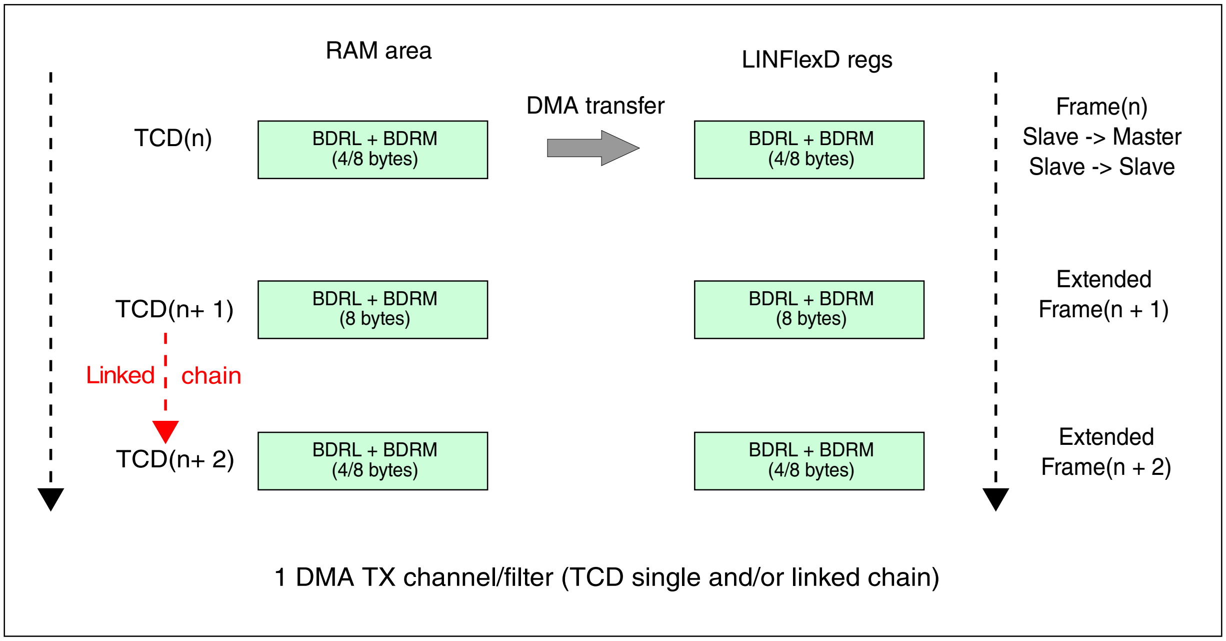


Figure 52-25. Slave node – TX memory map

The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in Table 52-11.

Table 52-11. Slave node – Tx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR

Slave to Master or Slave to Slave	DDRQ = 0
	DTRQ = 0
	HTRQ = 0
		To enable an ID filter (Tx mode) for each DMA TX channel	• Identifier list mode
			• Identifier mask mode
				DFL = payload size
				ID = address
				CCS = checksum
				DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1.The TCD setting (word transfer) is given in Table 52-12. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 52-12. TCD setting – Slave node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

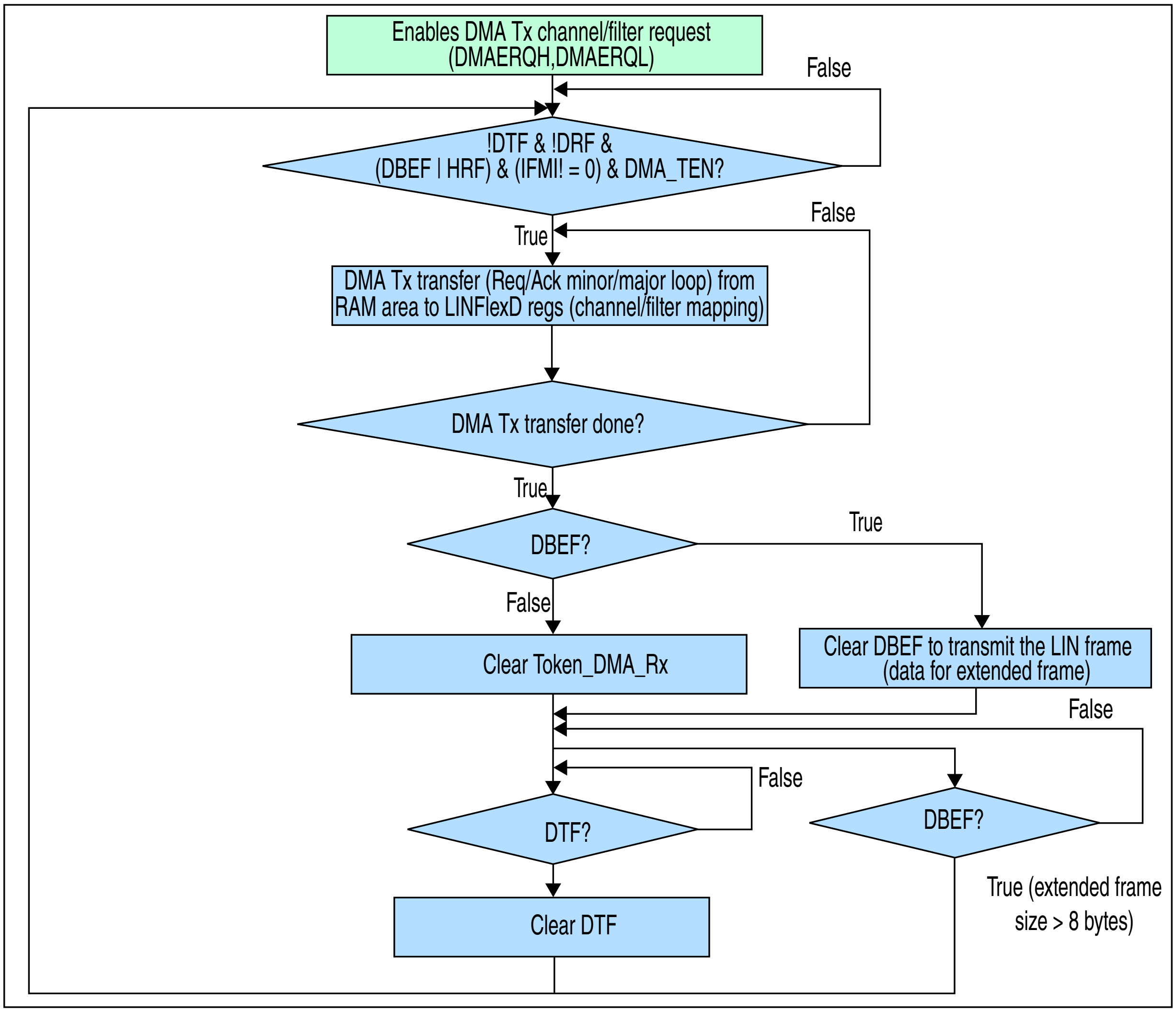

Enables DMA Tx channel/filter request (DMAERQH,DMAERQL)
False
!DTF & !DRF & (DBEF | HRF) & (IFMI! = 0) & DMA_TEN?
True
False
DMA Tx transfer (Req/Ack minor/major loop) from RAM area to LINFlexD regs (channel/filter mapping)
DMA Tx transfer done?
True
DBEF?
True
False
Clear Token_DMA_Rx
Clear DBEF to transmit the LIN frame (data for extended frame)
False
False
DTF?
DBEF?
Clear DTF
True (extended frame size > 8 bytes)

Figure 52-26. Slave node – DMA Tx FSM (concept scheme)

52.3.5.6 Slave node – RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

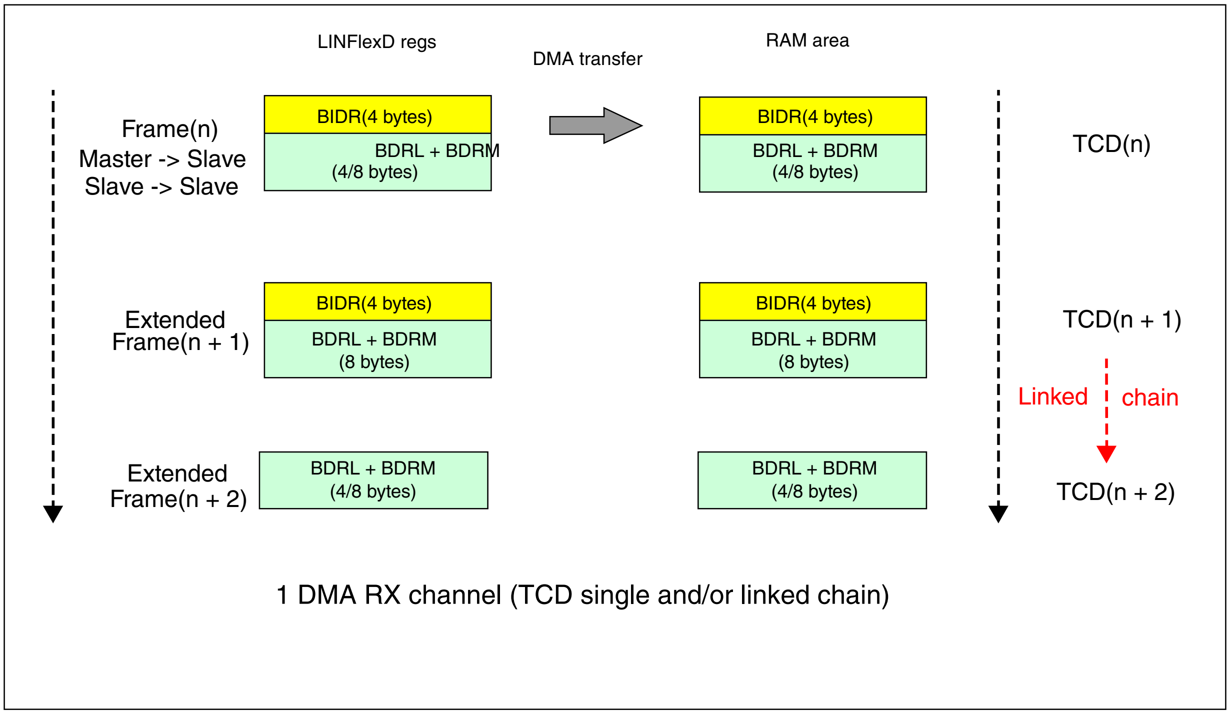


Figure 52-27. Slave node – RX memory map

The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, IFCR registers is given in Table 52-13.

Table 52-13. Slave node – Rx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR

Master to Slave or Slave to Slave	DDRQ = 0
	DTRQ = 0
	HTRQ = 0
		To enable an ID filter (Rx mode) for each DMA RX channel	• Identifier list mode
			• Identifier mask mode
				DFL = payload size
				ID = address
				CCS = checksum
				DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in the following figure. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in Table 52-14. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 52-14. TCD setting – Slave node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

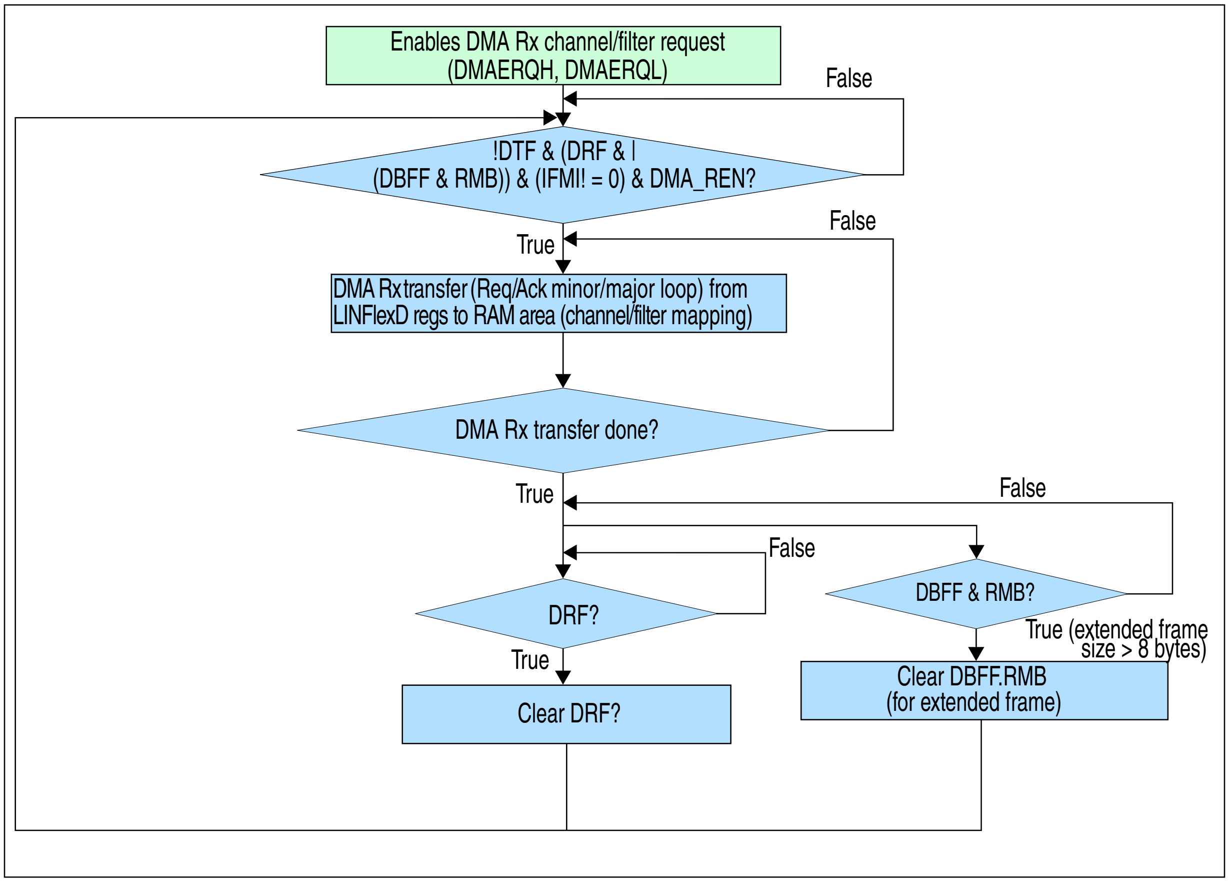

Enables DMA Rx channel/filter request
(DMAERQH, DMAERQL)
False
!DTF & (DRF & |
(DBFF & RMB)) & (IFMI! = 0) & DMA_REN?
True
False
DMA Rxtransfer (Req/Ack minor/major loop) from
LINFlexD regs to RAM area (channel/filter mapping)
DMA Rx transfer done?
True
False
False
DRF?
DBFF & RMB?
True (extended frame size > 8 bytes)
True
Clear DRF?
Clear DBFF.RMB
(for extended frame)

Figure 52-28. Slave node – DMA Rx FSM (concept scheme)

52.3.5.7 UART – TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Figure 52-29.

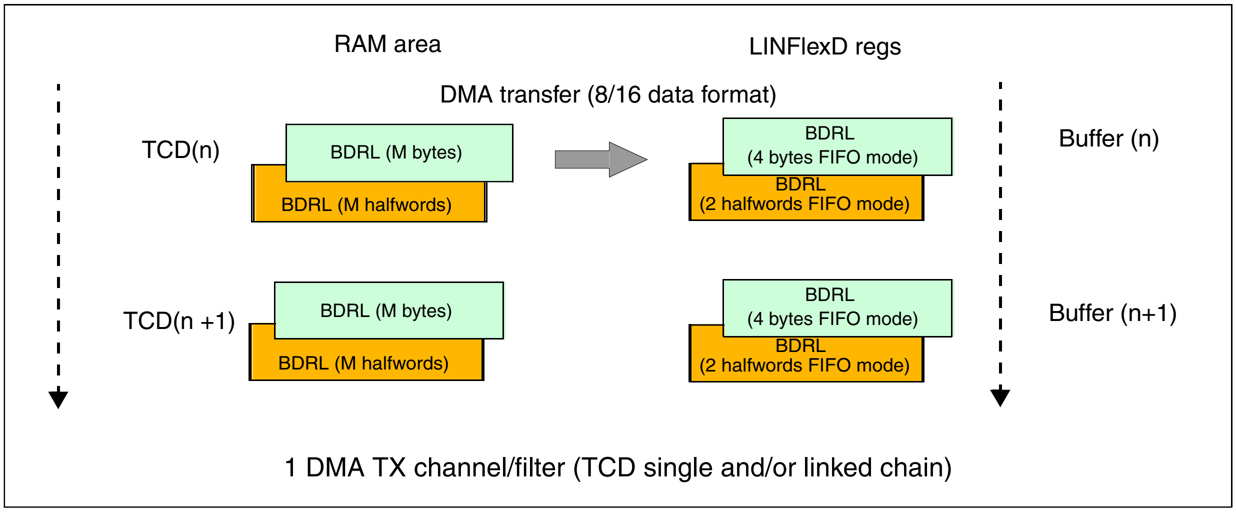


Figure 52-29. UART – TX memory map

The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8 bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

The concept FSM to control the DMA Tx interface is given in Figure 52-30. DMA Tx FSM will move to Idle state if DMATXE[0] = 0.The TCD setting (typical case) is given in Table 52-15. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 52-15. TCD setting – UART – Tx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		RAM address
SOFF[15:0]	1	2	Byte/Half-word increment
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	–M	–M × 2	-
DADDR[31:0]	-		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

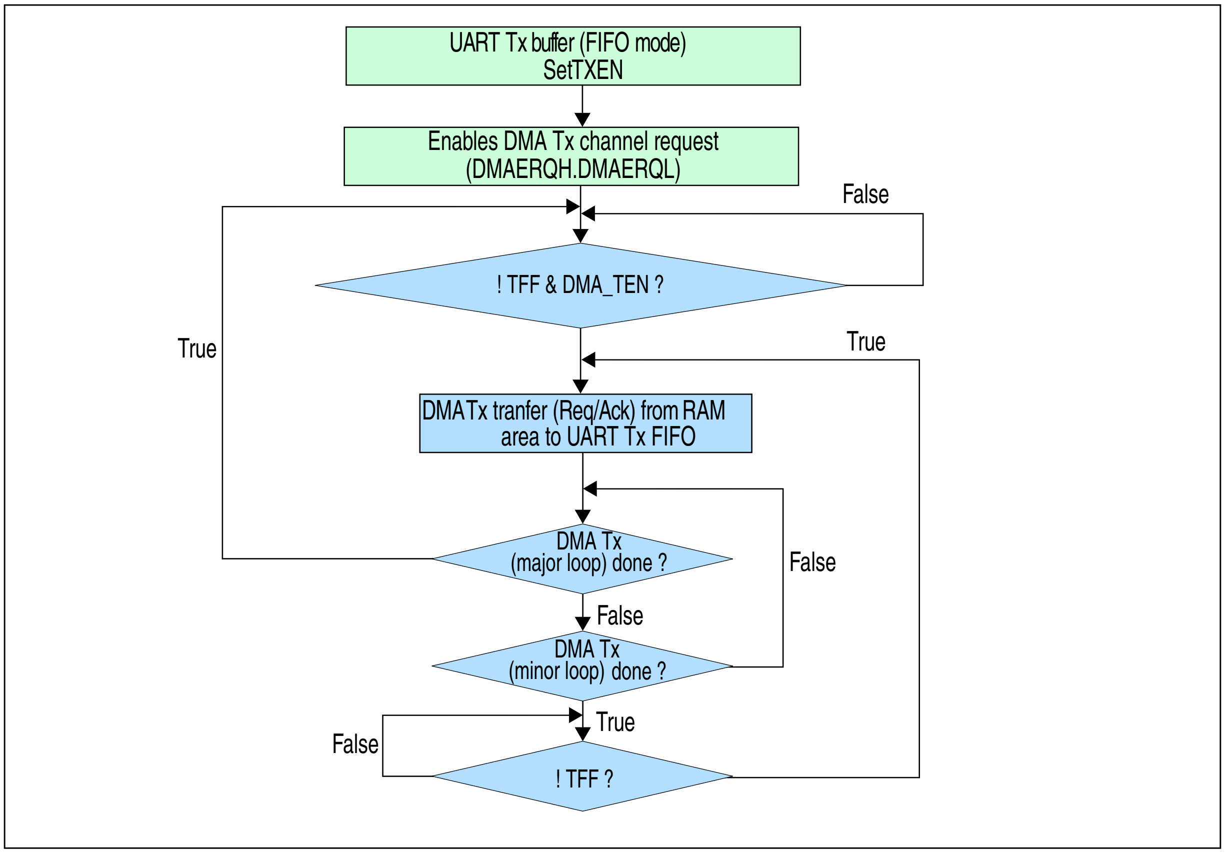


Figure 52-30. UART – DMA Tx FSM (concept scheme)

52.3.5.8 UART – RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Table 52-16.

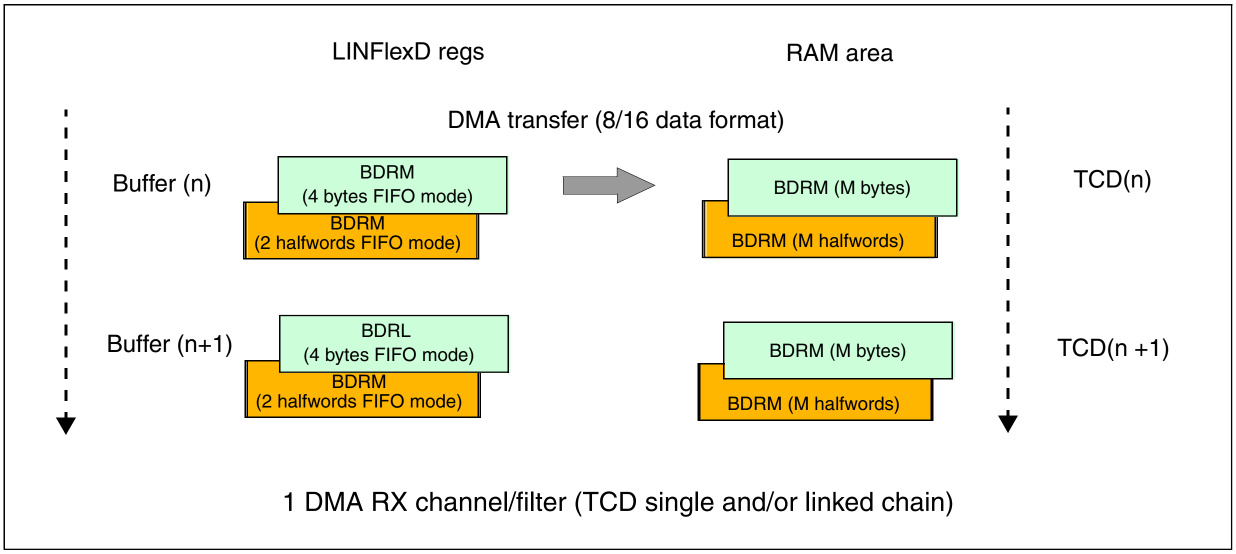


Figure 52-31. UART – RX memory map

The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format

This will sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

The concept FSM to control the DMA Rx interface is given in Figure 52-32. DMA Rx FSM will move to Idle state if DMARXE[0] = 0.The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/ halfword as soon an entry is available in the Rx FIFO. A new software reset bit is required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 52-16. TCD setting – UART – Rx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	0		
DADDR[31:0]	-		RAM address
DOFF[15:0]	1	2	Byte/Half-word increment
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	–M	–M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

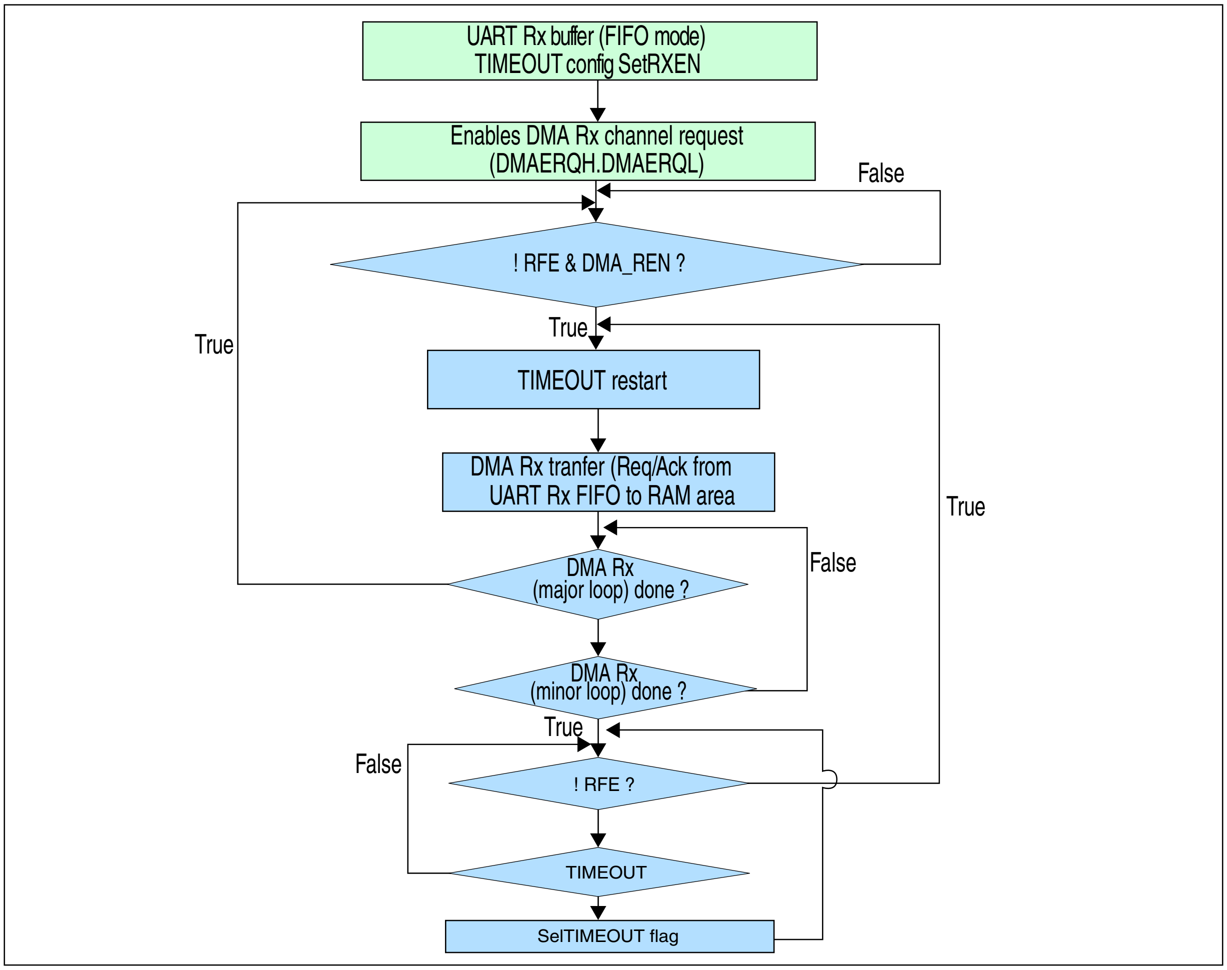


Figure 52-32. UART – DMA Rx FSM (concept scheme)

52.3.5.9 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index)
- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses

- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

52.4 Memory map and register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of generic slave = 0 and slave = 1.

NOTE

In Master mode, the registers IFCR0-IFCR15 are not present and the corresponding absolute address of the below register changes as shown in the following table.

Table 52-17. Offsets of the register from offsets 8C to 9C

Register name	Master/slave mode	Master only mode
LINFlexD_GCR	0x8C	0x4C
LINFlexD_UARTPTO	0x90	0x50
LINFlexD_UARTCTO	0x94	0x54
LINFlexD_DMATXE	0x98	0x58
LINFlexD_DMARXE	0x9C	0x5C

NOTE

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in transfer error.

NOTE

Any access to A0 location will not generate a transfer error.

LINFlexD memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LIN Control Register 1 (LINFlexD_LINCR1)	32	R/W	See section	52.4.1/1992
4	LIN Interrupt enable register (LINFlexD_LINIER)	32	R/W	0000_0000h	52.4.2/1995
8	LIN Status Register (LINFlexD_LINSR)	32	R/W	See section	52.4.3/1997
C	LIN Error Status Register (LINFlexD_LINESR)	32	w1c	0000_0000h	52.4.4/2001
10	UART Mode Control Register (LINFlexD_UARTCR)	32	R/W	0000_0000h	52.4.5/2002
14	UART Mode Status Register (LINFlexD_UARTSR)	32	R/W	0000_0000h	52.4.6/2008
18	LIN Time-Out Control Status Register (LINFlexD_LINTCSR)	32	R/W	0000_0200h	52.4.7/2010
1C	LIN Output Compare Register (LINFlexD_LINOCR)	32	R/W	0000_FFFFh	52.4.8/2012
20	LIN Time-Out Control Register (LINFlexD_LINTOCR)	32	R/W	See section	52.4.9/2013
24	LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)	32	R/W	0000_0000h	52.4.10/ 2013
28	LIN Integer Baud Rate Register (LINFlexD_LINIBRR)	32	R/W	0000_0000h	52.4.11/ 2014
2C	LIN Checksum Field Register (LINFlexD_LINCFR)	32	R/W	0000_0000h	52.4.12/ 2015
30	LIN Control Register 2 (LINFlexD_LINCR2)	32	R/W	See section	52.4.13/ 2016
34	Buffer Identifier Register (LINFlexD_BIDR)	32	R/W	0000_0000h	52.4.14/ 2018
38	Buffer Data Register Least Significant (LINFlexD_BDRL)	32	R/W	0000_0000h	52.4.15/ 2019
3C	Buffer Data Register Most Significant (LINFlexD_BDRM)	32	R/W	0000_0000h	52.4.16/ 2019
40	Identifier Filter Enable Register (LINFlexD_IFER)	32	R/W	0000_0000h	52.4.17/ 2020
44	Identifier Filter Match Index (LINFlexD_IFMI)	32	R	0000_0000h	52.4.18/ 2021
48	Identifier Filter Mode Register (LINFlexD_IFMR)	32	R/W	0000_0000h	52.4.19/ 2021
4C	Identifier Filter Control Register (LINFlexD_IFCR0)	32	R/W	0000_0000h	52.4.20/ 2022
50	Identifier Filter Control Register (LINFlexD_IFCR1)	32	R/W	0000_0000h	52.4.20/ 2022
54	Identifier Filter Control Register (LINFlexD_IFCR2)	32	R/W	0000_0000h	52.4.20/ 2022
58	Identifier Filter Control Register (LINFlexD_IFCR3)	32	R/W	0000_0000h	52.4.20/ 2022
5C	Identifier Filter Control Register (LINFlexD_IFCR4)	32	R/W	0000_0000h	52.4.20/ 2022
60	Identifier Filter Control Register (LINFlexD_IFCR5)	32	R/W	0000_0000h	52.4.20/ 2022
64	Identifier Filter Control Register (LINFlexD_IFCR6)	32	R/W	0000_0000h	52.4.20/ 2022

Table continues on the next page...

LINFlexD memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
68	Identifier Filter Control Register (LINFlexD_IFCR7)	32	R/W	0000_0000h	52.4.20/ 2022
6C	Identifier Filter Control Register (LINFlexD_IFCR8)	32	R/W	0000_0000h	52.4.20/ 2022
70	Identifier Filter Control Register (LINFlexD_IFCR9)	32	R/W	0000_0000h	52.4.20/ 2022
74	Identifier Filter Control Register (LINFlexD_IFCR10)	32	R/W	0000_0000h	52.4.20/ 2022
78	Identifier Filter Control Register (LINFlexD_IFCR11)	32	R/W	0000_0000h	52.4.20/ 2022
7C	Identifier Filter Control Register (LINFlexD_IFCR12)	32	R/W	0000_0000h	52.4.20/ 2022
80	Identifier Filter Control Register (LINFlexD_IFCR13)	32	R/W	0000_0000h	52.4.20/ 2022
84	Identifier Filter Control Register (LINFlexD_IFCR14)	32	R/W	0000_0000h	52.4.20/ 2022
88	Identifier Filter Control Register (LINFlexD_IFCR15)	32	R/W	0000_0000h	52.4.20/ 2022
8C	Global Control Register (LINFlexD_GCR)	32	R/W	0000_0000h	52.4.21/ 2023
90	UART Preset Timeout Register (LINFlexD_UARTPTO)	32	R/W	0000_0FFFh	52.4.22/ 2025
94	UART Current Timeout Register (LINFlexD_UARTCTO)	32	R	0000_0000h	52.4.23/ 2025
98	DMA Tx Enable Register (LINFlexD_DMATXE)	32	R/W	0000_0000h	52.4.24/ 2026
9C	DMA Rx Enable Register (LINFlexD_DMARXE)	32	R/W	0000_0000h	52.4.25/ 2027

52.4.1 LIN Control Register 1 (LINFlexD_LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

NOTE

> When accessing the LINCR1 register, each reserved bit should be written to its original reset value.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD	CFD	LASE	AUTOWU	MBL				BF	0	LBKM	MME	SSBL	RBLM	SLEEP	INIT
W										▨						
Reset	0	0	0	0	0	0	0	0	1	0	0	*	0	0	1	0

* Notes:
- MME field: If slave = 0, then MME bit is 1, else MME bit is 0

LINFlexD_LINCR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CCD	Checksum Calculation disable This bit can be written during Initialization mode only. It is read-only in Normal mode. 0 Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read only. 1 Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to sent a software calculated checksum/CRC (provided CFD is reset).
17 CFD	Checksum field disable This bit can be configured during Initialization mode only. It is read-only in Normal mode. 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame
18 LASE	LIN Autosynchronization Enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0). 0 Autosynchronization disabled 1 Autosynchronization enabled
19 AUTOWU	Auto Wakeup This bit can be configured during Initialization mode only. This bit is utilized in UART mode also. 0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set
20–23 MBL	Master Break Length These bits choose the length of the Sync break to be generated by the master. These bits can be programmed in Initialization mode only. They are read-only in Normal mode. 0000 10-bit break length 0001 11-bit break length 0010 12-bit break length 0011 13-bit break length 0100 14-bit break length 0101 15-bit break length 0110 16-bit break length

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
	0111 17-bit break length 1000 18-bit break length 1001 19-bit break length 1010 20-bit break length 1011 21-bit break length 1100 22-bit break length 1101 23-bit break length 1110 36-bit (cooling) break length 1111 50-bit break length
24 BF	By-pass filter This bit can be programmed during Initialization mode only. **NOTE:** If generic slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed. 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBKM	Loop Back mode Refer to "Loop back mode" in Test mode . Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also. 0 Loop Back Mode disabled 1 Loop Back mode enabled
27 MME	Master mode enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. **NOTE:** If generic slave = 0, then this bit will read a '1' always, and cannot be programmed. 0 Slave Mode 1 Master Mode
28 SSBL	Slave Mode Sync Break Length This bit can be programmed in Initialization mode only. It is read only in Normal mode. 0 11 bit break length 1 10 bit break length
29 RBLM	Receiver Buffer Locked mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also. 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun. Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also.

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also.

52.4.2 LIN Interrupt enable register (LINFlexD_LINIER)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0		FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIETOIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINIER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZIE	Stuck at zero Interrupt Enable An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR.

Table continues on the next page...

LINFlexD_LINIER field descriptions (continued)

Field	Description
	0 No interrupt 1 Interrupt enabled
20 HEIE	Header Error Interrupt Enable An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. **NOTE:** If generic slave = 0, then this bit will always read a 0 and cannot be programmed. 0 No interrupt 1 Interrupt enabled
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR. 0 No interrupt 1 Interrupt enabled
25 LSIE	LIN state Interrupt enable Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag. 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum
26 WUIE	Wakeup interrupt enable If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated. 0 No interrupt 1 Interrupt enabled
27 DBFIE	Data Buffer Full Interrupt enable An interrupt is generated if this bit is set and the DBFF bit in LINSR is also set. 0 No interrupt 1 Interrupt enabled
28 DBEIETOIE	Data Buffer Empty Interrupt enable/Timeout Interrupt Enable An interrupt is generated if this bit is set and LINSR[DBEF] status bit is set (in LIN mode) or if this bit is set and UARTSR[TO] status bit is set (in UART mode). 0 No interrupt 1 Interrupt enabled
29 DRIE	Data Reception complete Interrupt enable An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled

Table continues on the next page...

LINFlexD_LINIER field descriptions (continued)

Field	Description
30 DTIE	Data Transmitted Interrupt enable An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
31 HRIE	Header Received Interrupt An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed. 0 No interrupt 1 Interrupt enabled

52.4.3 LIN Status Register (LINFlexD_LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												AUTOSYNC_COMP	RDC		
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS				0		RMB	DRBNE	RXbusy	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
W	1						w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	*	0	0	0	0	0	0

* Notes:

- RDI field: Reset value of RDI reflects the RX pin state

LINFlexD_LINSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 AUTOSYNC_ COMP	AUTOSYNC_COMP This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers can be read in autosynchronization mode. **NOTE:** If autosynchronization is disabled, this bit is reserved.
13–15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table continues on the next page...

LINFlexD_LINSR field descriptions (continued)

Field	Description
16–19 LINS	LIN state In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits. **NOTE:** The value of this bit field doesn't change by any write operation to it. Writing 0xF to this bit field clears the Rx interrupt, only when set due to the LIN state event. 0000 Sleep mode: LINFlexD is in Sleep mode, to save power consumption. 0001 Init mode: This mode is entered when: SLEEP bit and INIT bit are reset by software; Wakeup pulse has been received on RX pin (AUTOWU set); Previous frame transmission/reception has been completed/aborted 0010 Idle mode: This mode is entered when: SLEEP bit and INIT bit are reset by software; Wakeup pulse has been received on RX pin (AUTOWU set); Previous frame transmission/reception has been completed/aborted 0011 Sync break: In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. Note: In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle. 0100 Sync Del: In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 0101 Sync Field: In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 0110 Identifier Field: In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 0111 Header Reception/Transmission: In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 1000 Data Reception/Data Transmission: In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 1001 Checksum: Data transmission/reception completed, checksum transmission/reception ongoing.
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.
24 RXbusy	Receiver Busy flag In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit. 0 Receiver is idle 1 Reception ongoing

Table continues on the next page...

LINFlexD_LINSR field descriptions (continued)

Field	Description
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **NOTE:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin. 1. When slave is in Sleep mode 2. When master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27 DBFF	Data Buffer full flag This bit is set by hardware when the buffer is full (8 bytes received) and DFL > 7. It should be cleared by software. This bit is also reset by hardware in Initialization mode.
28 DBEF	Data Buffer empty flag This bit is set by hardware when the buffer is empty (8 bytes have been transmitted) and DFL > 7. It should be cleared by software after data buffer is refilled by software. Transmission of next 8 bytes will not start unless this bit is reset by software. This bit is also reset by hardware in Initialization mode.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** In case framing error or checksum error occurs then this flag is not set.
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** For LIN mode, in case a bit error occurs (and IOBE is reset) then this flag is not set.
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: a) All filters are inactive and Bypass filter (BF) bit is set b) No match in any filter and Bypass filter (BF) bit is set c) TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

52.4.4 LIN Error Status Register (LINFlexD_LINESR)

Refer to Errors for detailed description of all errors.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	BEF	CEF	SFEF	SDEF	IDPEF	FEF	BOF	0						NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c							w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINESR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0: No output compare event occurred 1: In master mode, LINESR[OCF] flag is set when counter LINTCSR[CNT] has matched the content of LINOCR[OC2]. In slave mode, LINESR[OCF] is set when the content of the counter LINTCSR[CNT] has matched the content of LINOCR[OC1] or LINOCR[OC2]. This bit should be cleared by software. If this bit is set, MODE bit in LINTCSR is cleared, and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.

Table continues on the next page...

LINFlexD_LINESR field descriptions (continued)

Field	Description
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINCR1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. **NOTE:** Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set. If generic slave = 0, then this bit will always read a 0, and cannot be programmed. For generic slave = 1,this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.
25–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

52.4.5 UART Mode Control Register (LINFlexD_UARTCR)

NOTE

When accessing the UARTCR register, reserved bits should always be written to zero.

NOTE

- The LINFlexD module in UART mode does not support communication with Special Word Length (UARTCR[WLS] = 1) in buffer mode.
- The LINFlexD module in UART mode supports communication with Special Word Length (UARTCR[WLS] = 1) in FIFO mode only with UARTCR[WL1, WL0] = 1,1.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0								NEF			DTU_PCETX	SBUR		WLS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TDFL_TFC			RDFL_RFC			RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE	WL0	UART
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTCR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received. Register bit can be read in any mode, written only in Initialization mode.
12 DTU_PCETX	Disable Timeout in UART mode This bit can be programmed in Initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received. **In Buffer mode:** This timer gets reset whenever the number of bytes received (rec_byte_cnt) is equal to RDFL. At this point the DRF bit is also set and num_of_frames (number of frames received), rec_byte_cnt are reset. When num_of_frames equals NEF, Disable Timeout is generated. But this clears num_of_frames also. Thus the timer restarts again. The value of counter is 0 when it restarts. **In FIFO mode:**

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	When num_of_frames equals NEF, Disable Timeout is generated which clears num_of_frames and causes the timer to restart again. The value of counter is 0 when it restarts. Notes: • Disable Timeout causes only the Timer to restart which enables Timeout again. • Timer reset means it resets and starts counting again and the Timeout is enabled. 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received
13–14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. This bit can be programmed in the initialization mode only, when the UART bit is set. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 reserved
15 WLS	Special Word Length in UART mode This bit can be programmed in initialization mode only when the UART bit is set. If this bit is set, setting WL and PCE bits has no effect in UART reception although parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode. 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support.
16–18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter TDFL defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL is a read/write field. Bit 15 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations TDFL = x01 or TDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes Register bit is a read/write field. TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TFCx is a read-only field available for debug purposes. 000 : empty 001 : 1 byte 010 : 2 bytes 011 : 3 bytes 100 : 4 bytes Others : reserved TDFLTFC can be programmed and are significative only when the UART bit is set.

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	Register bit can be read in any mode, written only in Initialization mode. **NOTE:** When a debugger is connected, this counter will decrement if TxFIFO is being read through the debugger. **NOTE:** In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.
19–21 RDFL_RFC	Reception Data Field Length /RX FIFO Counter RDFL defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations RDFL = x01 or RDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RFCx is a read-only field available for debug purposes. 000 : Empty 001 : 1 byte 010 : 2 bytes/1.5 byte in case of WLS bit setting 011 : 3 bytes 100 : 4 bytes/2x1.5 byte in case of WLS bit setting Others - Reserved RDFLRFC can be programmed and are significative only when the UART bit is set. **NOTE:** In buffer mode, RDFL should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint. **NOTE:** When a debugger is connected, this counter will decrement if RxFIFO is being read through the debugger. **NOTE:** In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.
22 RFBM	RFBM Rx Fifo/Buffer mode This bit can be programmed in Initialization mode, only when the UART bit is set. Register bit can be read in any mode, written only in initialization mode. 0 Rx Buffer mode enabled 1 Rx Fifo mode enabled (mandatory in DMA Rx mode)
23 TFBM	Tx Fifo/Buffer mode This bit can be programmed in initialization mode, only when the UART bit is set. Register bit can be read in any mode, written only in initialization mode. 0 Tx Buffer mode enabled 1 Tx Fifo mode enabled (mandatory in DMA Tx mode)
24 WL1	Word Length in UART mode This bit can be programmed in Initialization mode only, when the UART bit is set. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>WL1</th><th>WL0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>7 bits data + parity</td></tr>
<tr><td>0</td><td>1</td><td>8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1</td></tr>
<tr><td>1</td><td>0</td><td>15 bits data + parity</td></tr>
<tr><td>1</td><td>1</td><td>16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1</td></tr>
</table>
</td></tr>
<tr><td>25
PC1</td><td>Parity Control
This bit can be programmed in Initialization mode, only when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Parity sent is Even</td></tr>
<tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table>
</td></tr>
<tr><td>26
RxEn</td><td>Receiver Enable
This bit can be programmed only when the UART bit is set.
0 Receiver disabled
1 Receiver enabled</td></tr>
<tr><td>27
TxEn</td><td>Transmitter Enable
This bit can be programmed only when UART bit is set.
0 Transmitter disabled
1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed</td></tr>
<tr><td>28
PC0</td><td>Parity Control
This bit can be programmed in Initialization mode, only when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Parity sent is Even</td></tr>
<tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>29
PCE</td><td>Parity Control Enable
This bit can be programmed in Initialization mode only, when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Parity transmit/check Disable
1 Parity transmit/check Enable</td></tr>
<tr><td>30
WL0</td><td>Word Length in UART mode
This bit can be programmed in Initialization mode only, when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>WL1</th><th>WL0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>7 bits data + parity</td></tr>
<tr><td>0</td><td>1</td><td>8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1</td></tr>
<tr><td>1</td><td>0</td><td>15 bits data + parity</td></tr>
<tr><td>1</td><td>1</td><td>16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1</td></tr>
</table>
</td></tr>
<tr><td>31
UART</td><td>UART Mode
This bit can be programmed in Initialization mode only.
Register bit can be read in any mode, written only in initialization mode.
0 LIN mode
1 UART mode</td></tr>
</table>

52.4.6 UART Mode Status Register (LINFlexD_UARTSR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE				RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRFRFE	DTFTFF	NF
W	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c		w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTSR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
17 OCF	Output Compare Flag An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. **NOTE:** For Baud rate above 1 Mbit/s, this flag is not usable. 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCR

Table continues on the next page...

LINFlexD_UARTSR field descriptions (continued)

Field	Description
18–21 PE	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. **NOTE:** Parity is checked only after complete frame is received. Following are the conditions when WL bits = 01 or 11 (either PE0 or PE2 is only set). • When PE0 is set, it indicates Parity error in either first or second byte received • When PE2 is set, it indicates Parity error in either third or fourth byte received 0 No parity error 1 Parity error in the corresponding received byte
22 RMB	Release Message Buffer This bit should be cleared by software. This bit will reflect the same value as in LINSR, when in Initialization mode and the UART bit is set. 0 Buffer data is free 1 Buffer data ready to be read by software
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the new value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software. The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode.

Table continues on the next page...

LINFlexD_UARTSR field descriptions (continued)

Field	Description
	An interrupt will be generated when LINIER.DBEIE/TOIE bit is set on the Error interrupt line in UART mode.
29 DRFRFE	Data Reception Completed Flag /Rx FIFO Empty Flag DRF is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF bit is set when the configured number of valid stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF is set irrespective of framing error in case framing error is in the last STOP bit configured, parity error or overrun error. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTFTFF	Data Transmission Completed Flag/ TX FIFO Full Flag DTF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.

52.4.7 LIN Time-Out Control Status Register (LINFlexD_LINTCSR)

This register contains control and status bits for timeout feature.

NOTE

When the MODE bit is 0, any activity on the transmit or receive pins will cause an unwanted change in the value of the 8-bit field Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR). The LINFlexD module is enabled in LIN mode and the value of the MODE bit of the LIN Timeout Control Status register (LINTCSR) is changed from ‘1’ to ‘0’, then the old value of the Output Compare Value 1 (OC1) and Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR) is retained. As a consequence, if the module is reconfigured from UART to LIN mode, or LINTCSR MODE bit is changed from ‘1’ to ‘0’, an incorrect timeout exception is generated when a LIN communication starts. To avoid this, set mode bit to '1', then reconfigure the LINFlexD in LIN mode and then reset the MODE bit. Before switching LINTCSR[MODE] from 1 to 0 in between frames, load LINOCR with 0xFFFF.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					MODE	IOT	TOCE	CNT							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

LINFlexD_LINTCSR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MODE	Time-out counter mode This bit can be configured only during initialization. Register bit can be read in any mode, written only in initialization mode. **NOTE:** Always set LINTCSR[MODE] to 1 when UARTCR[UART] is 1. **NOTE:** Before switching LINTCSR[MODE] from 1 to 0 in between frames, load LINOCR to 0xFFFF. 0 LIN mode 1 Output compare mode
22 IOT	Idle on timeout Register bit can be read in any mode, written only in initialization mode. This feature is applicable only when MODE bit in LINTCSR is cleared.

Table continues on the next page...

LINFlexD_LINTCSR field descriptions (continued)

Field	Description
	0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event
23 TOCE	Time-out counter enable TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE. 0 Time-out counter disable. OCF flag is not set on an output compare event. 1 Time-out counter enable. OCF flag is set if an output compare event occurs.
24–31 CNT	Counter Value These bits reflect the value of a counter used for timeout. **NOTE:** For proper functionality of this counter, LINIBRR should be >= 5.

52.4.8 LIN Output Compare Register (LINFlexD_LINOCR)

This register contains the value to be compared to the LINTCSR:CNT value. This register is writable by software only in Output Compare Mode.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																OC2								OC1							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_LINOCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 OC2	Output compare value 2
24–31 OC1	Output compare value 1

52.4.9 LIN Time-Out Control Register (LINFlexD_LINTOCR)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				RTO				0	HTO						
W																
Reset	0	0	0	0	1	1	1	0	0	*	*	*	*	*	*	*

* Notes:

- HTO field: HTO reset values:

  HTO resets to 0011100b in Master only mode, if generic slave = 0.

  HTO resets to 0101100b in Master/Slave mode, if generic slave = 1.

LINFlexD_LINTOCR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 RTO	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0Eh = 14, corresponding to T $_{\text{Response\_Maximum}}$ = 1.4 x T $_{\text{Response\_Nominal}}$
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 HTO	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. If Master mode is enabled then these bits will always reflect 28 (1.4 x 10 bits of sync + 1.4 x 10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

52.4.10 LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												FBR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINFBRR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 FBR	Fractional Baud rates Register bit can be read in any mode, written only in initialization mode. 0000 Fraction(LDIV) = 0 0001 Fraction(LDIV) = 1/16 0010 Fraction(LDIV) = 2/16 0011 Fraction(LDIV) = 3/16 0100 Fraction(LDIV) = 4/16 0101 Fraction(LDIV) = 5/16 0110 Fraction(LDIV) = 6/16 0111 Fraction(LDIV) = 7/16 1000 Fraction(LDIV) = 8/16 1001 Fraction(LDIV) = 9/16 1010 Fraction(LDIV) = 10/16 1011 Fraction(LDIV) = 11/16 1100 Fraction(LDIV) = 12/16 1101 Fraction(LDIV) = 13/16 1110 Fraction(LDIV) = 14/16 1111 Fraction(LDIV) = 15/16

52.4.11 LIN Integer Baud Rate Register (LINFlexD_LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBRR. It can be programmed only in Initialization mode.

NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												IBR																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINIBRR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12–31
IBR
	Integer Baud rates
	These bits along with the fractional baud rate bits decide the LIN baud rate.
	IBR = 0h: LIN clock disabled
	IBR = 1h: Mantissa (LDIV) = 1
	...
	IBR = FFFFEh: Mantissa (LDIV) = 1048574
	IBR = FFFFFh: Mantissa (LDIV) = 1048575
	Register bit can be read in any mode, written only in initialization mode.

52.4.12 LIN Checksum Field Register (LINFlexD_LINCFR)

NOTE

> There is a delay between 4 to 6 clock cycles of PBRIDGEx_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.

This register consists of checksum bits.

CFD	CCD	LINCHKSUM Read/write	Checksum sent
1	1	read/write	None
1	0	read-only	None
0	1	read/write	Programmed checksum
0	0	read-only	Calculated checksum

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								CF							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINCFR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 CF	Checksum bits When the CCD bit is reset these bits are read-only and are calculated by hardware. When the CCD bit is set, these bits can be written by software.

52.4.13 LIN Control Register 2 (LINFlexD_LINCR2)

This register includes control status bits related to buffer operations.

NOTE

When accessing the LINCR2 register, each reserved bit should be written to its original reset value.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBDE	IOBE	IOPE	WURQ	DDRQ	DTRQ	ABRQ	HTRQ	0							
W																
Reset	0	1	*	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- IOPE field: When slave = 0, this field always reads ‘0’ and cannot be programmed else this bit is programmable and reset value is 1.

LINFlexD_LINCR2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TBDE	Two Bit delimiter bit This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits
17 IOBE	Idle on Bit Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine
18 IOPE	Idle on Identifier Parity Error This bit can be set in Initialization mode only.

Table continues on the next page...

LINFlexD_LINCR2 field descriptions (continued)

Field	Description
	Register bit can be read in any mode, written only in initialization mode. If generic slave = 0, then this bit will always read a 1 else 0, and cannot be programmed. 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request. Register bit can be read/set by software
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered. Register bit can be read/set by software
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete. Register bit can be read/set by software
22 ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also. Register bit can be read/set by software Register bit can be read/set by software
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header. Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. **NOTE:** In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.4.14 Buffer Identifier Register (LINFlexD_BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

NOTE

All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL						DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BIDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL = Number of data bytes - 1 **NOTE:** DFL[5:3] is provided for extended frames. Normally LIN mode will use DFL[2:0]. Identifier filters are now compatible with DFL[5:0] also.
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copy them in the BDR registers 1 LINFlexD transmits the data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data filed only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = '1').

52.4.15 Buffer Data Register Least Significant (LINFlexD_BDRL)

This register is a part of an 8-byte data buffer.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA3								DATA2								DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BDRL field descriptions

Field	Description
0–7 DATA3	Data Byte 3 Data byte 3 of the data field.
8–15 DATA2	Data Byte 2 Data byte 2 of the data field.
16–23 DATA1	Data Byte 1 Data byte 1of the data field.
24–31 DATA0	Data Byte 0 Data byte 0 of the data field.

52.4.16 Buffer Data Register Most Significant (LINFlexD_BDRM)

This register is a part of an 8-byte data buffer.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA7								DATA6								DATA5								DATA4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BDRM field descriptions

Field	Description
0–7 DATA7	Data Byte 7 Data byte 7 of the data field.
8–15 DATA6	Data Byte 6 Data byte 6 of the data field.

Table continues on the next page...

LINFlexD_BDRM field descriptions (continued)

Field	Description
16–23 DATA5	Data Byte 5 Data byte 5of the data field.
24–31 DATA4	Data Byte 4 Data byte 4 of the data field.

52.4.17 Identifier Filter Enable Register (LINFlexD_IFER)

This register enables/disables a particular filter.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																FACT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 FACT	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. Register bit can be read in any mode, written only in initialization mode. 1. In Identifier mask mode, FACT (2n+1) have no effect on the corresponding filters as they act as mask for the Identifier 2n. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. **NOTE:** Refer to chip configuration details to see the number of filters used in the device.

52.4.18 Identifier Filter Match Index (LINFlexD_IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											IFMI				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFMI field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IFMI	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see Slave mode for more details).

52.4.19 Identifier Filter Mode Register (LINFlexD_IFMR)

This register configures the modes of filters.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								IFM							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFMR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 IFM	Filter mode Register bit can be read in any mode, written only in initialization mode. 0 Filters 2n and 2n+1 are in identifier list mode 1 Filters 2n and 2n+1 are in mask mode, where filter 2n+1 is the mask for filter 2n

52.4.20 Identifier Filter Control Register (LINFlexD_IFCR*n*)

This register is read-only in normal mode and can be programmed only in Initialization mode.

Even-numbered instances (IFCR0, IFCR2, IFCR4, ...) of this register act as filters in both Identifier list and Identifier mask modes.

Odd-numbered instances (IFCR1, IFCR3, IFCR5, ...) of this register act:

- As filters in Identifier list mode
- As a mask for the preceding-numbered register in identifier mask mode

Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL						DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFCR*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes - 1 Register bit can be read in any mode, written only in initialization mode.
22 DIR	Direction This bit controls the direction of the data field. Register bit can be read in any mode, written only in initialization mode. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_IFCR*n* field descriptions (continued)

Field	Description
	0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. Register bit can be read in any mode, written only in initialization mode.

52.4.21 Global Control Register (LINFlexD_GCR)

This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

The address offset depends on the no_of_filters. Refer to the chip configuration details for the number of filters used in this device.

Table 52-18. Register fields reset by SR

Register	Comment
LINSR	All fields except RXbusy and AUTOSYNC_COMP are reset
LINESR	All fields are reset
LINTCSR	Only CNT[0:7] is reset
UARTSR	All fields except RFE & TFF are reset
UARTCR	Only TFC & RFC are reset
UARTCTO	All fields are reset

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										TDFBM	RDFBM	TDLIS	RDLIS	STOP	SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_GCR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of received data is LSB — in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB — in other words, the first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Received data is not inverted 1 Received data is inverted
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). Register bit can be read in any mode, written only in initialization mode. 0 1 stop bit 1 2 stop bits
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). Register bit can be written only be software in initialization mode. Bit is always read 0 by software. Table 52-18 describes the register fields reset by SR.

52.4.22 UART Preset Timeout Register (LINFlexD_UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be received. This register can be written by software any time.

The address offset depends on no_of_filters. Refer to the chip configuration details to see the number of filters used in the device.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																				PTO											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_UARTPTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 PTO	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer also to register UARTCTO.

52.4.23 UART Current Timeout Register (LINFlexD_UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see UART Preset Timeout Register (LINFlexD_UARTPTO) to monitor the number of bits received by UART. UART timeout works in both CPU and DMA modes.

The timeout counter:

- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGEx_CLK
- Is automatically enabled when UARTCR[RXEN] = 1

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO

- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																				CTO											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTCTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 CTO	Current Timeout CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

52.4.24 DMA Tx Enable Register (LINFlexD_DMATXE)

This register enables the DMA TX interface. This register can be written and read by software anytime.

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																DTE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_DMATXE field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved

Table continues on the next page...

LINFlexD_DMATXE field descriptions (continued)

Field	Description
	NOTE: The number of reserved bits varies, and is equal to 32 - $2^{TX\_CH\_NUM}$. These lies from 0 to (31 - $2^{TX\_CH\_NUM}$). Refer to the chip configuration details for the value of TX_CH_NUM used in this device.
16–31 DTE	DMA Tx channel Y enable **NOTE:** The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. The size is [$2^{TX\_CH\_NUM}$ - 1 : 0]. The position of these bits are (32 - $2^{TX\_CH\_NUM}$) to 31. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of TX_CH_NUM used in this device. 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled

52.4.25 DMA Rx Enable Register (LINFlexD_DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

The address offset depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																DRE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_DMARXE field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{RX\_CH\_NUM}$. These lies from 0 to (31 - $2^{RX\_CH\_NUM}$). Refer to the chip configuration details for the value of RX_CH_NUM used in this device.
16–31 DRE	DMA Rx channel Y enable **NOTE:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. The size is [$2^{RX\_CH\_NUM}$ -1 : 0]. The position of these bits are (32 - $2^{RX\_CH\_NUM}$) to 31. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of RX_CH_NUM used in this device. 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled

52.5 Programming considerations

The next subsections describe the various configurations in which the LINFlexD module can be used.

52.5.1 Master node

LINFlexD acts as Master when the MME bit is set.

52.5.1.1 Transmitter

Transmitter Master Node - Transmitter configuration.

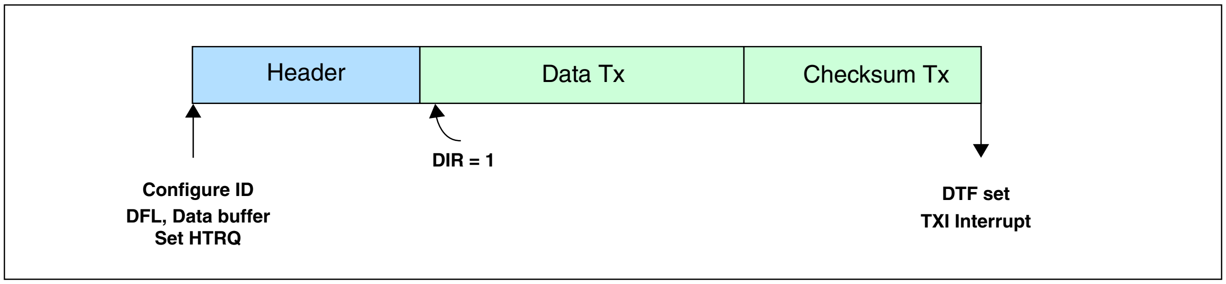


Figure 52-33. Master node – Transmitter configuration

52.5.1.2 Receiver

Receiver Master Mode - RX Configuration

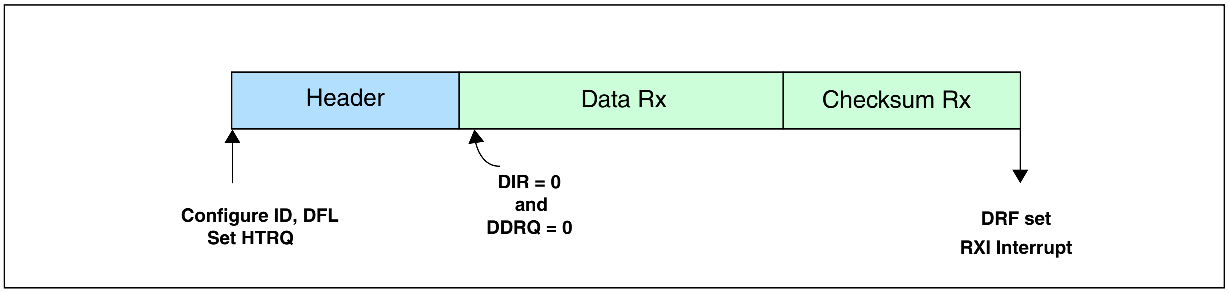


Figure 52-34. Master node – Receiver configuration

52.5.1.3 Transmitter, Bit Error

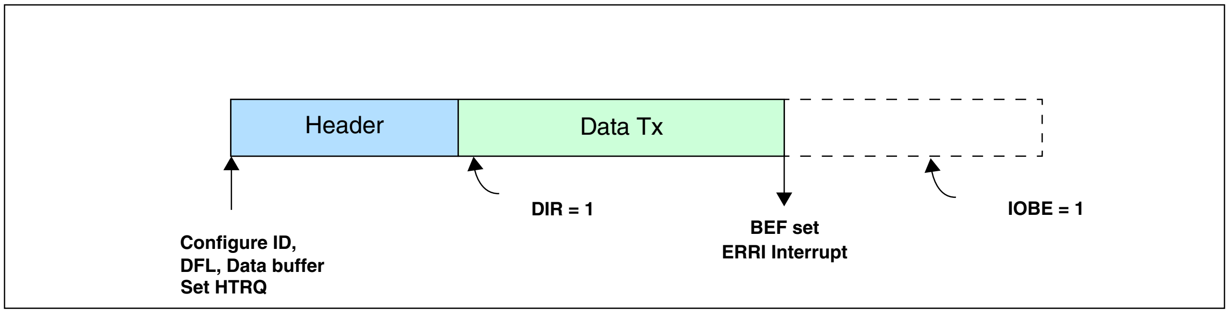


Figure 52-35. Master node – Transmitter, Bit Error configuration

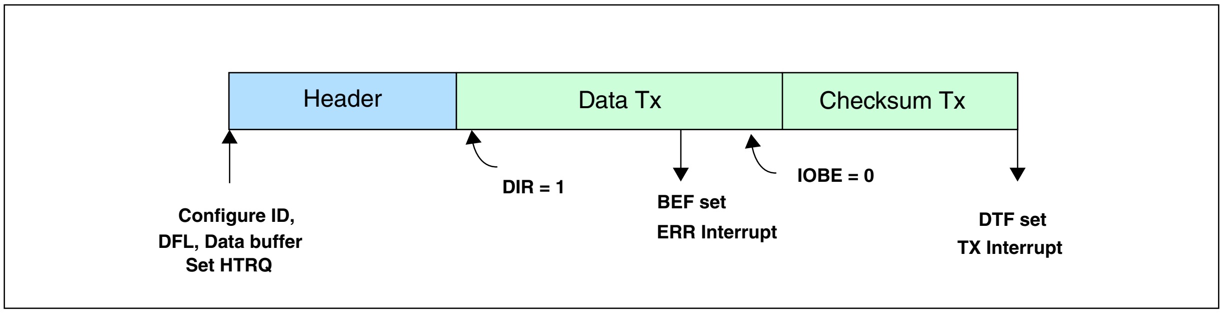


Figure 52-36. Master node – Transmitter, Checksum Error configuration

52.5.1.4 Receiver, Checksum Error

Checksum Error for Receiver.

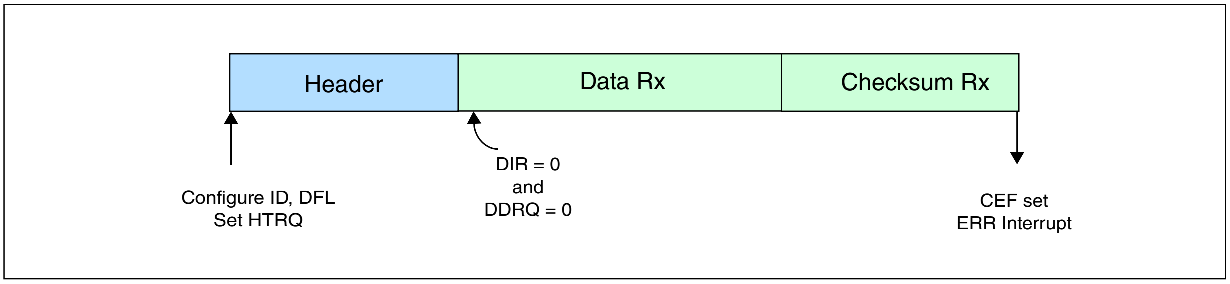


Figure 52-37. Master node – Receiver, Checksum Error configuration

52.5.2 Slave node

Slave node (transmitter).

52.5.2.1 Transmitter (no identifier filters)

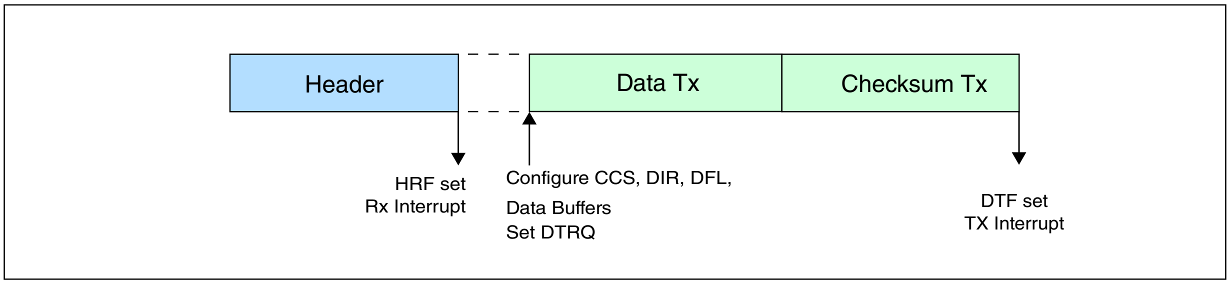


Figure 52-38. Slave node – Transmitter (no identifier filters) configuration

52.5.2.2 Receiver (no identifier filters)

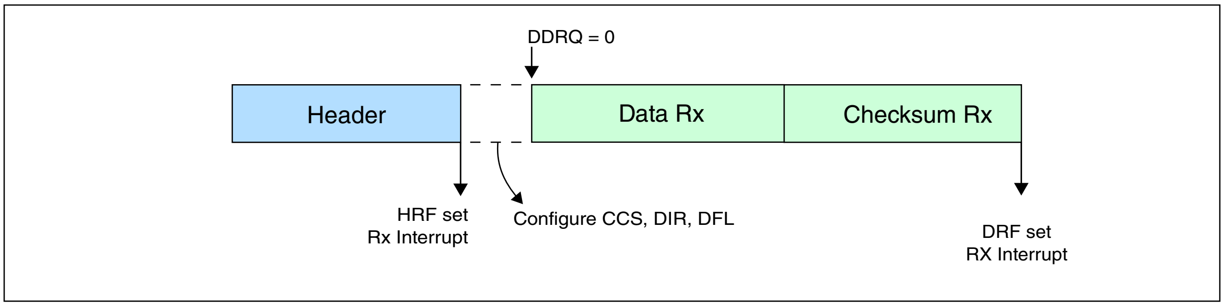


Figure 52-39. Slave node – Receiver (no identifier filters) configuration

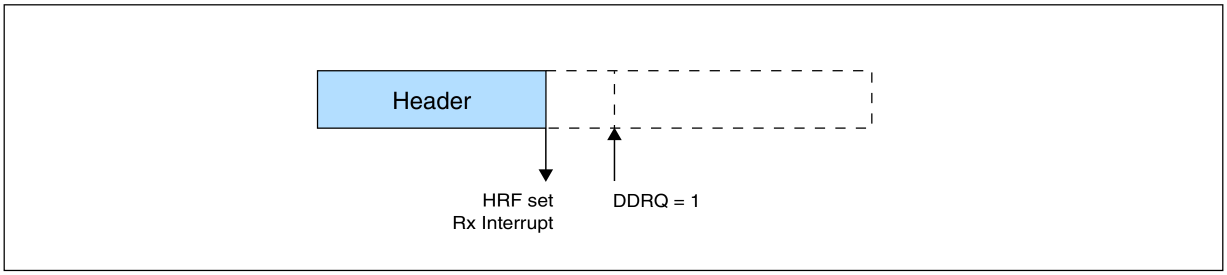


Figure 52-40. Slave node – Receiver (no identifier filters) configuration

52.5.2.3 No filters, Transmitter, Bit error

The following figure shows Slave node - No filters, Transmitter, Bit error configuration.

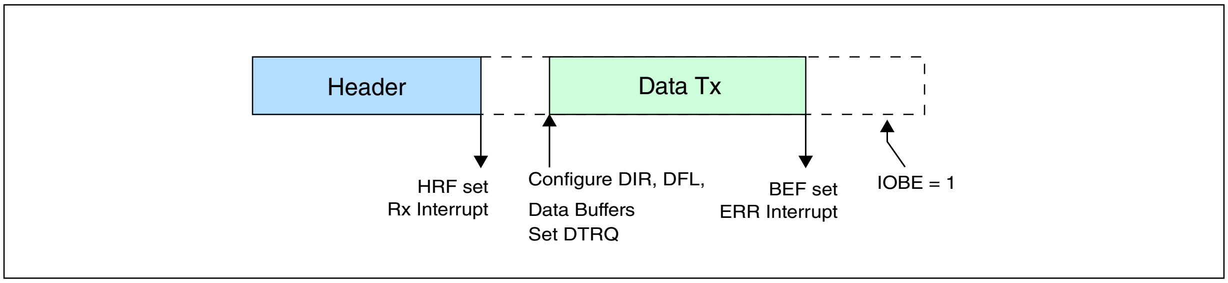


Figure 52-41. Slave node – No filters, Transmitter, Bit error configuration

52.5.2.4 No filters, Receiver, Checksum Error

The following figure shows Slave node - No filters, Receiver, Checksum error configuration.

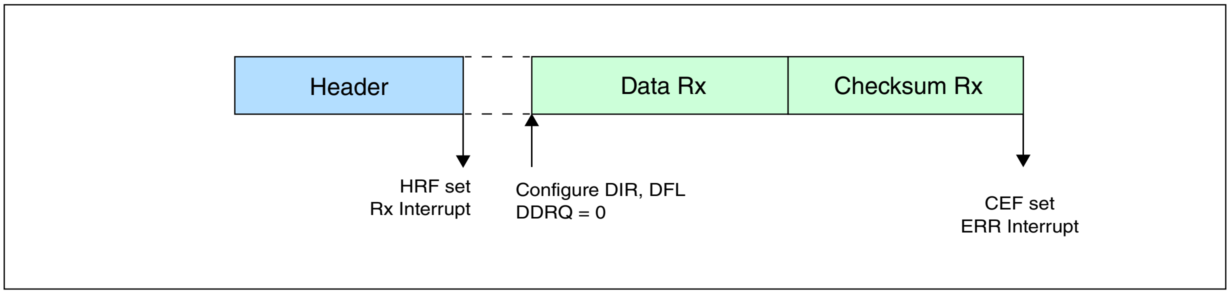


Figure 52-42. Slave node – No filters, Receiver, Checksum error configuration

52.5.2.5 At least one TX filter, BF is reset, ID matches filter

This configuration can be used in case slave never receives data, for example, sensor.

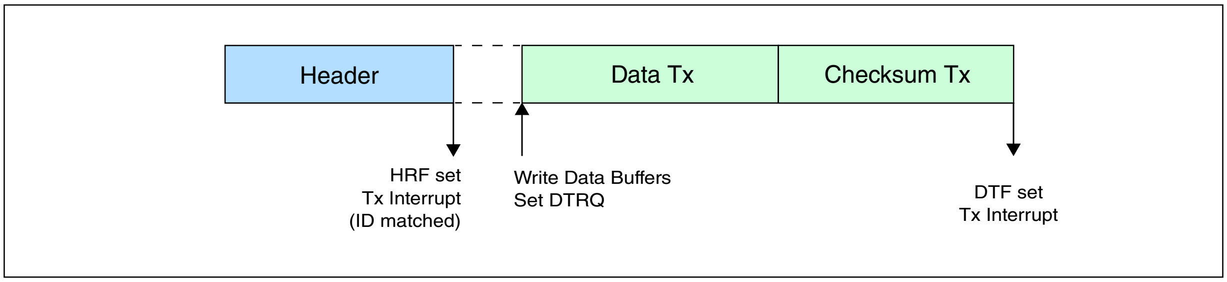


Figure 52-43. Slave node – one TX filter configuration

52.5.2.6 At least one RX filter, BF reset, ID matches filter

The following figure shows Slave node - one RX filter configuration.

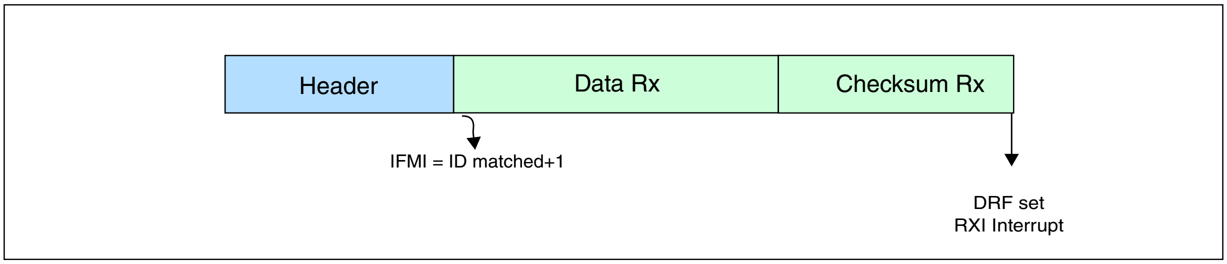


Figure 52-44. Slave node – one RX filter configuration

52.5.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

The following figure shows RX only, TX only, RX and TX filters configuration.

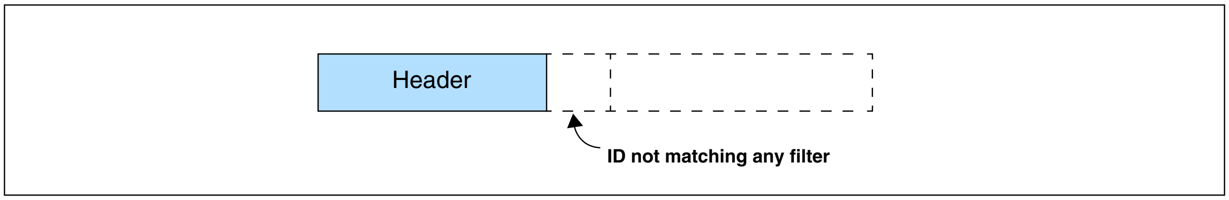


Figure 52-45. RX only, TX only, RX & TX filters configuration

52.5.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

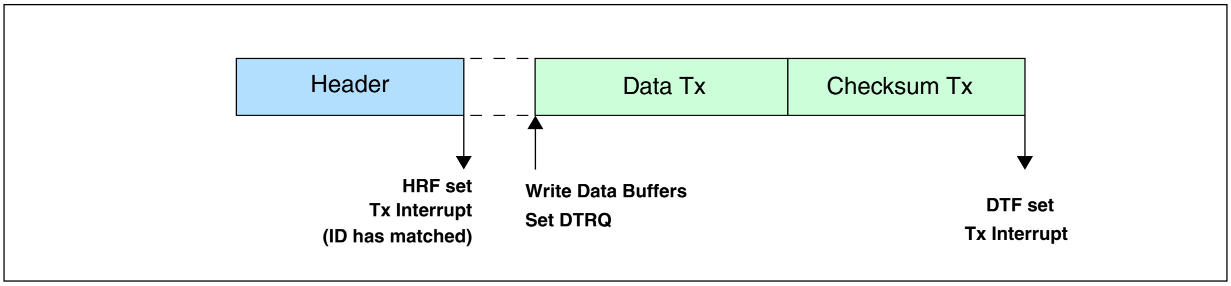


Figure 52-46. TX filter, BF is set configuration – ID has matched

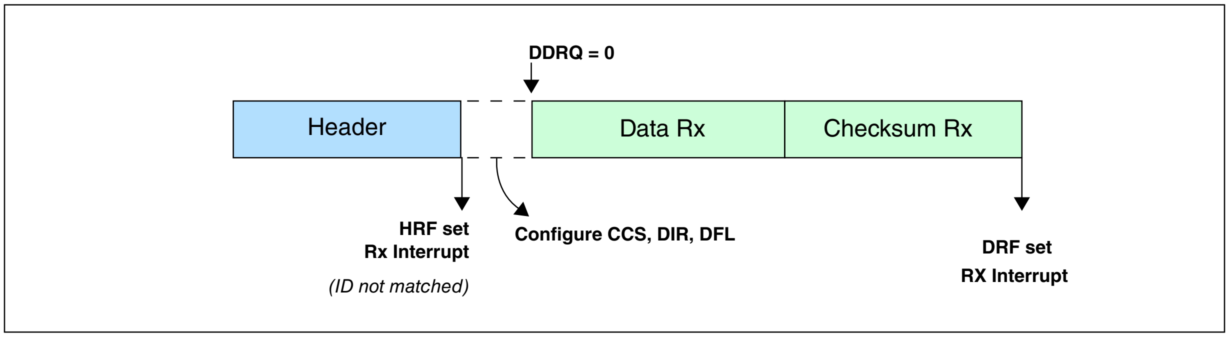


Figure 52-47. TX filter, BF is set configuration – ID not matched

52.5.2.9 RX filter, BF is set

The following figures show RX filter configurations.

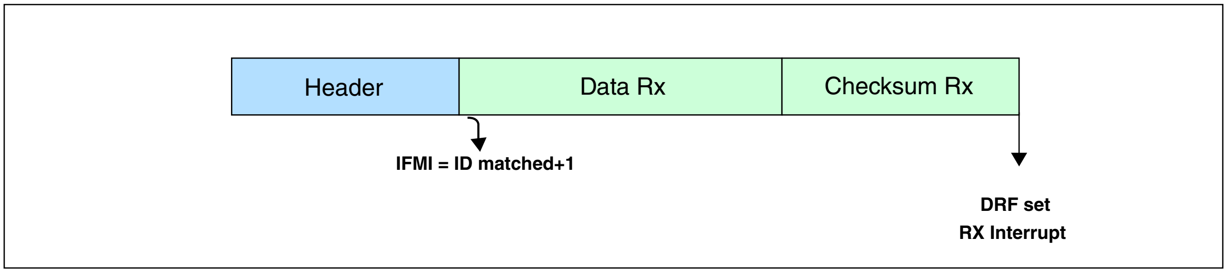


Figure 52-48. RX filter, BF is set configuration – ID matched

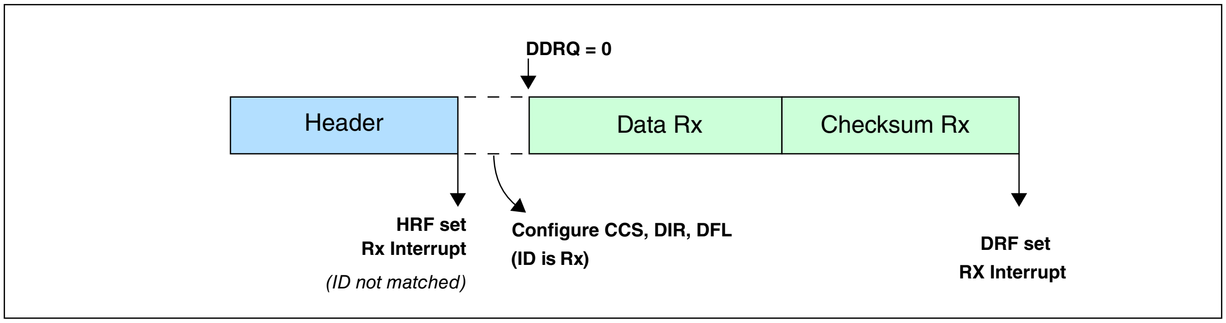


Figure 52-49. RX filter, BF is set configuration – ID not matched (ID is Rx)

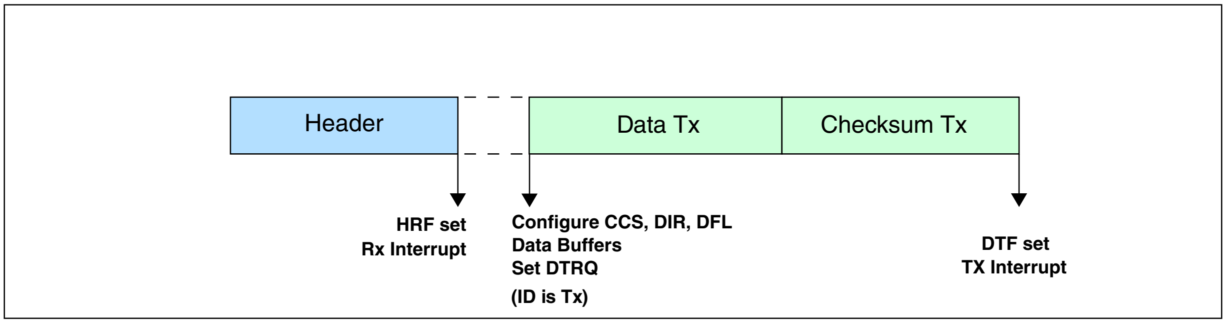


Figure 52-50. RX filter, BF is set configuration – (ID is Tx)

52.5.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

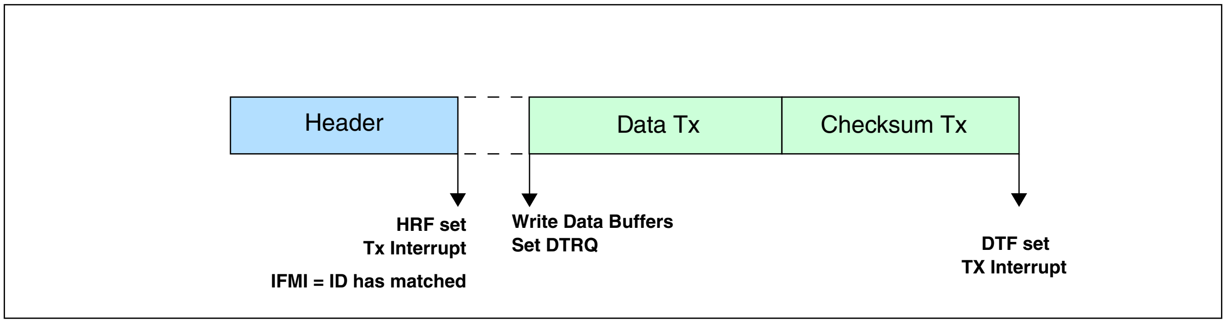


Figure 52-51. TX filter, RX filter, BF set configuration – (ID matched)

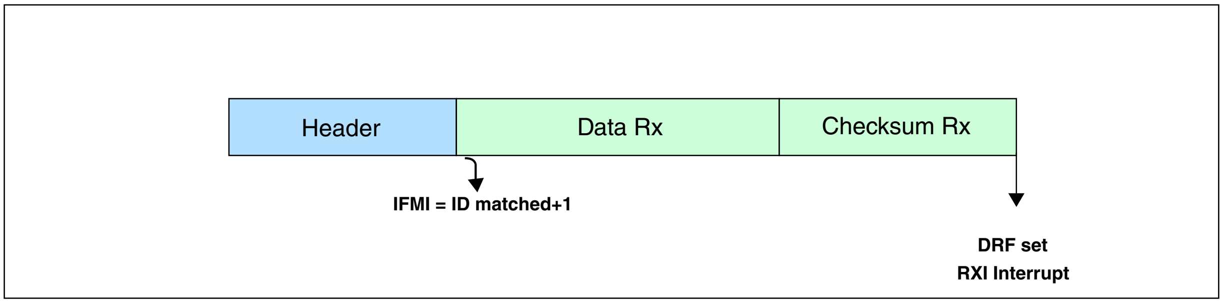


Figure 52-52. TX filter, RX filter, BF set configuration – (ID matched + 1)

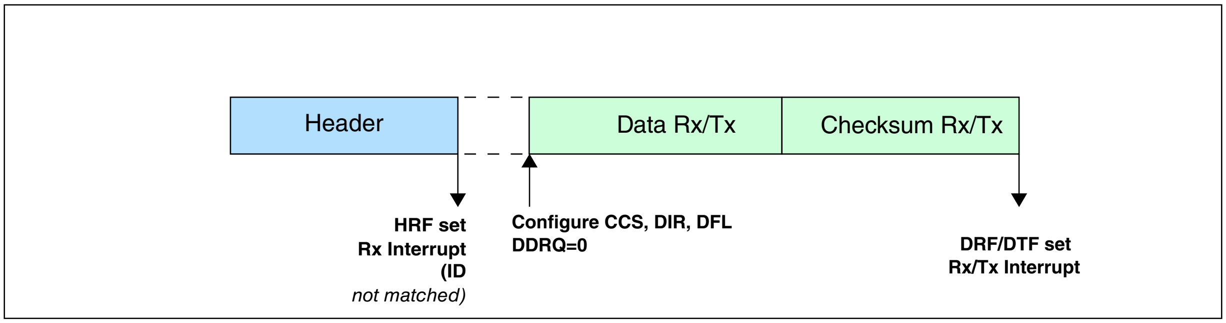


Figure 52-53. TX filter, RX filter, BF set configuration – (ID not matched, RX/TX Interrupt)

52.5.3 Extended frames

The following figure shows Extended frames (RX Interrupt).

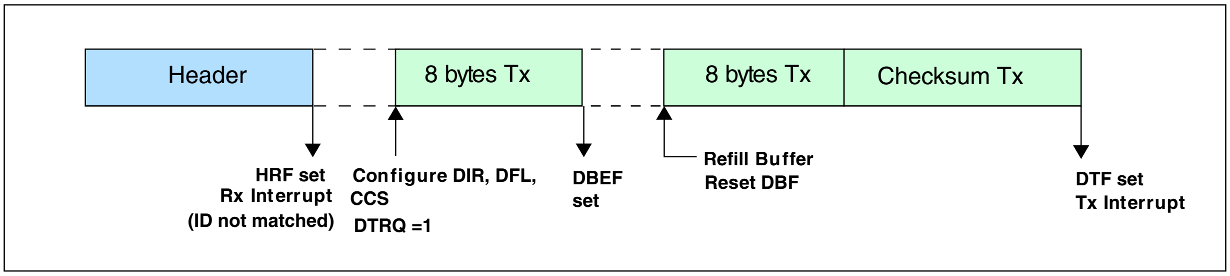


Figure 52-54. Extended frames (TX Interrupt)

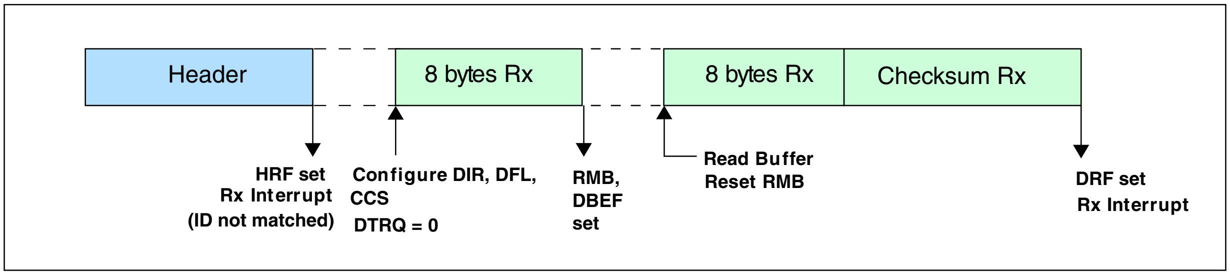


Figure 52-55. Extended frames (RX Interrupt)

52.5.4 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

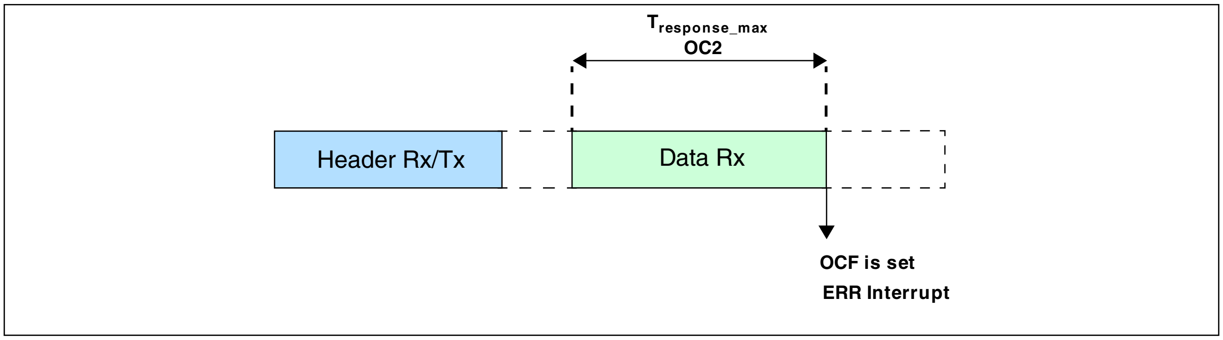


Figure 52-56. Response timeout

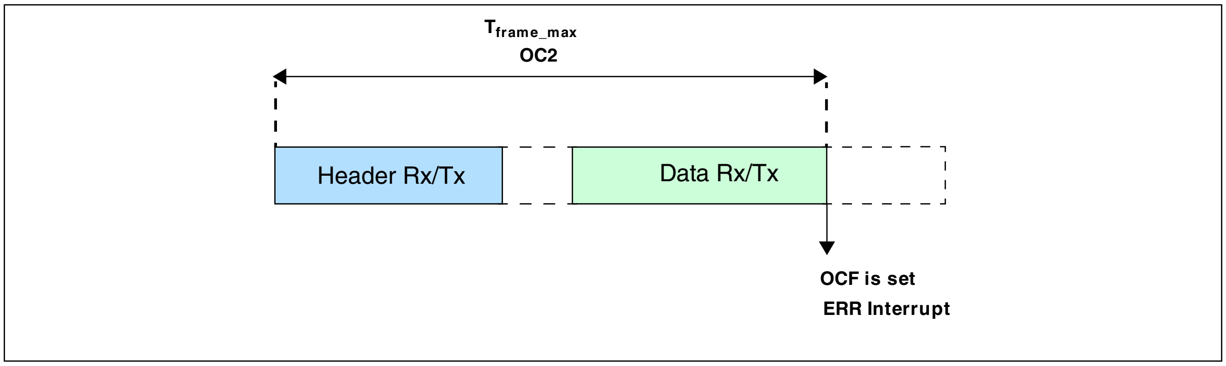


Figure 52-57. Frame timeout

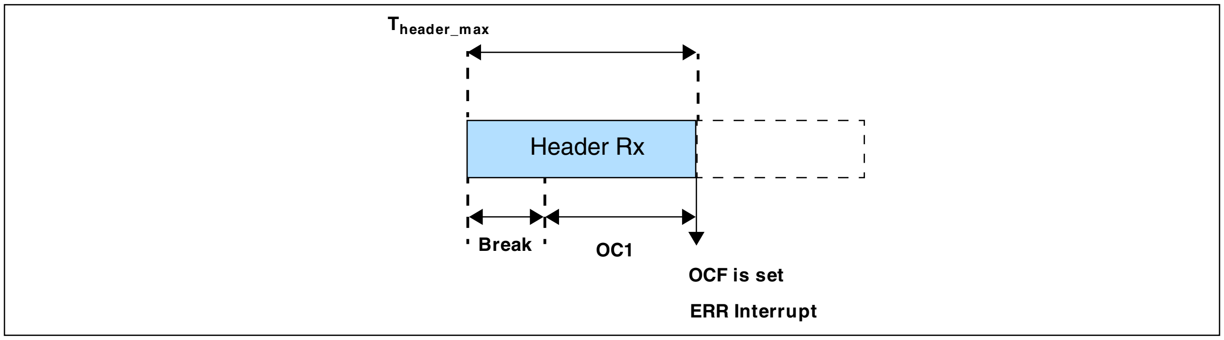


Figure 52-58. Header timeout

52.5.5 UART mode

The following figure shows UART mode configuration.

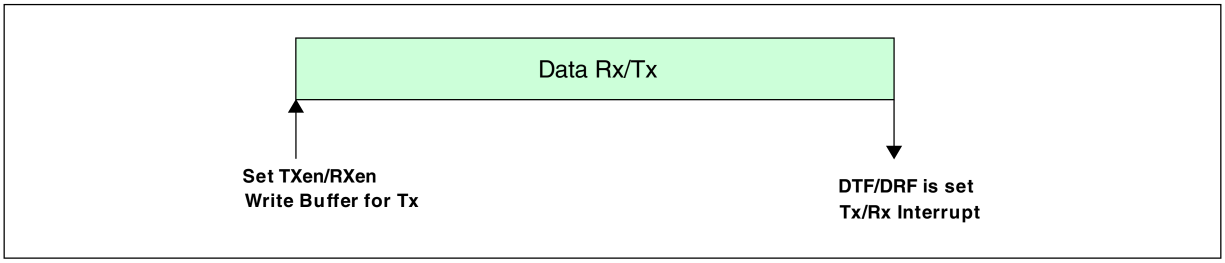


Figure 52-59. UART mode configuration

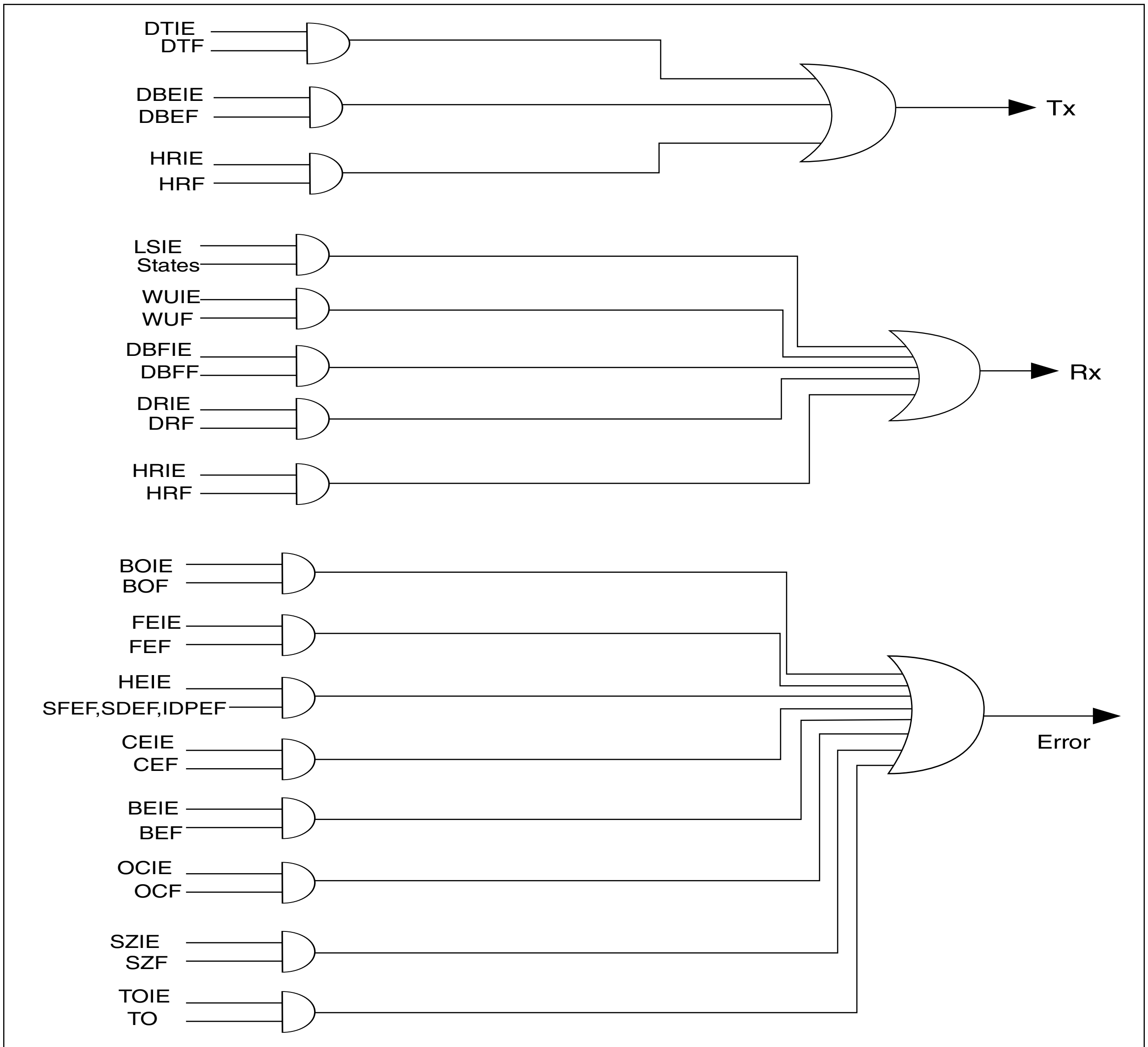
52.5.6 Interrupts

Interrupt section.

Table 52-19. Interrupts

Interrupt Event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error
Header error	SFEF orSDEF orIDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer Overrun error	BOF	BOIE	Error
UART Timeout error	TO	TOIE	Error
Lin state	Sync Del, Sync Field,Identifier field or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data buffer full	DBFF	DBFIE	Rx
Data buffer empty	DBEF	DBEIE	Tx
Data Reception Complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

The following figure shows an Interrupt Flow Diagram.



Note: Rx interrupt due to LIN state event is cleared by writing LINSR[LINS] field with 0xF when LSIE is enabled

Figure 52-60. Interrupt diagram

52.5.7 LINFlexD Clock Tolerance

1. **Faster receiver tolerance:**
   In this case the receiver has a higher baud rate than the transmitter, thus the stop bit sampling starts already in the last transmitted payload bit. To ensure the correct noise and framing error free reception bit , the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

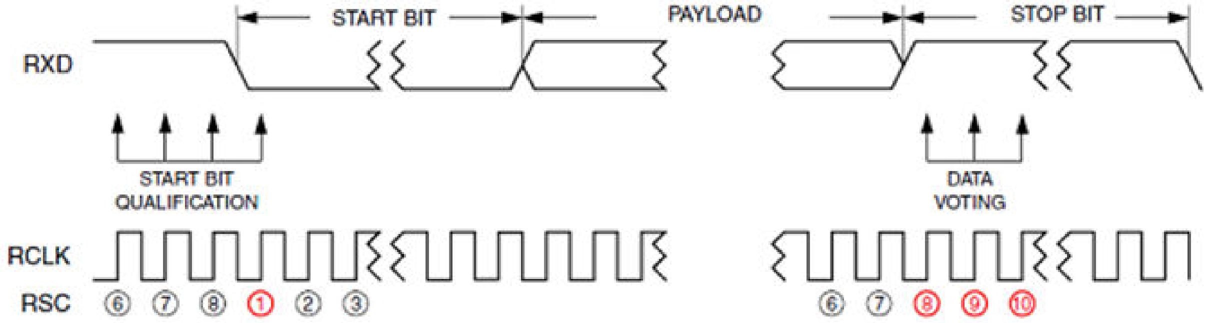


Figure 52-61. Faster receiver

2. **Slower receiver tolerance:**
   In this case the receiver has a slower baud rate than the transmitter, thus the stop bit sampling is still running while the next start bit is already transmitted. To ensure the correct noise and framing error free reception bit , the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

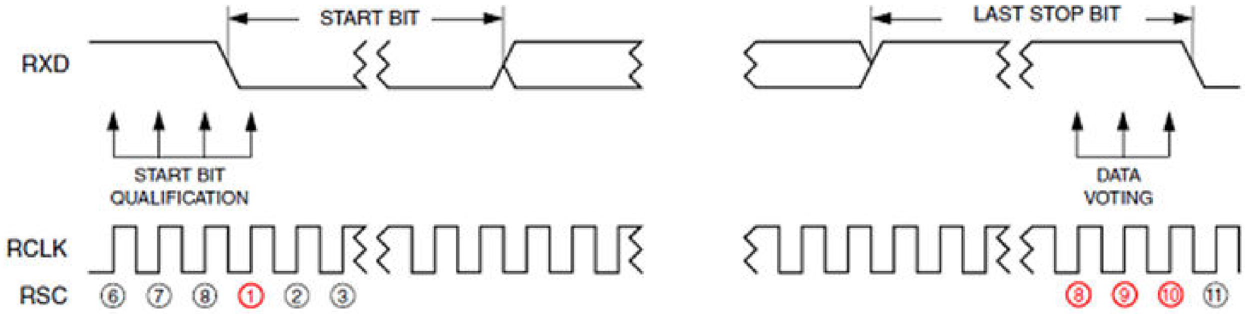


Figure 52-62. Slower receiver

Chapter 53
10/100-Mbps Ethernet MAC (ENET)

53.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The MAC-NET core, in conjunction with a 10/100-Mbit/s MAC, implements layer 3 network acceleration functions. These functions are designed to accelerate the processing of various common networking protocols, such as IP, TCP, UDP, and ICMP, providing wire speed services to client applications.

53.2 Overview

The core implements a dual-speed 10/100-Mbit/s Ethernet MAC compliant with the IEEE802.3-2002 standard. The MAC layer provides compatibility with half- or full-duplex 10/100-Mbit/s Ethernet LANs.

The MAC operation is fully programmable and can be used in Network Interface Card (NIC), bridging, or switching applications. The core implements the remote network monitoring (RMON) counters according to IETF RFC 2819.

The core also implements a hardware acceleration block to optimize the performance of network controllers providing TCP/IP, UDP, and ICMP protocol services. The acceleration block performs critical functions in hardware, which are typically implemented with large software overhead.

The core implements programmable embedded FIFOs that can provide buffering on the receive path for lossless flow control.

Advanced power management features are available with magic packet detection and programmable power-down modes.

A unified DMA (uDMA), internal to the ENET module, optimizes data transfer between the ENET core and the SoC, and supports an enhanced buffer descriptor programming model to support IEEE 1588 functionality.

The programmable Ethernet MAC with IEEE 1588 integrates a standard IEEE 802.3 Ethernet MAC with a time-stamping module. The IEEE 1588 standard provides accurate clock synchronization for distributed control nodes for industrial automation applications.

53.2.1 Features

53.2.1.1 Ethernet MAC features

- Implements the full 802.3 specification with preamble/SFD generation, frame padding generation, CRC generation and checking
- Supports zero-length preamble
- Dynamically configurable to support 10/100-Mbit/s operation
- Supports 10/100 Mbit/s full-duplex and configurable half-duplex operation
- Compliant with the AMD magic packet detection with interrupt for node remote power management
- Seamless interface to commercial ethernet PHY devices via one of the following:
    - a 4-bit Media Independent Interface (MII) operating at 2.5/25 MHz.
    - a 4-bit non-standard MII-Lite (MII without the CRS and COL signals) operating at 2.5/25 MHz.
    - a 2-bit Reduced MII (RMII) operating at 50 MHz.
- Simple 64-Bit FIFO user-application interface
- CRC-32 checking at full speed with optional forwarding of the frame check sequence (FCS) field to the client
- CRC-32 generation and append on transmit or forwarding of user application provided FCS selectable on a per-frame basis
- In full-duplex mode:
    - Implements automated pause frame (802.3 x31A) generation and termination, providing flow control without user application intervention
    - Pause quanta used to form pause frames — dynamically programmable
    - Pause frame generation additionally controllable by user application offering flexible traffic flow control
    - Optional forwarding of received pause frames to the user application
    - Implements standard flow-control mechanism
- In half-duplex mode: provides full collision support, including jamming, backoff, and automatic retransmission
- Supports VLAN-tagged frames according to IEEE 802.1Q

- Programmable MAC address: Insertion on transmit; discards frames with mismatching destination address on receive (except broadcast and pause frames)
- Programmable promiscuous mode support to omit MAC destination address checking on receive
- Multicast and unicast address filtering on receive based on 64-entry hash table, reducing higher layer processing load
- Programmable frame maximum length providing support for any standard or proprietary frame length
- Statistics indicators for frame traffic and errors (alignment, CRC, length) and pause frames providing for IEEE 802.3 basic and mandatory management information database (MIB) package and remote network monitoring (RFC 2819)
- Simple handshake user application FIFO interface with fully programmable depth and threshold levels
- Provides separate status word for each received frame on the user interface providing information such as frame length, frame type, VLAN tag, and error information
- Multiple internal loopback options
- MDIO master interface for PHY device configuration and management supports two programmable MDIO base addresses, and standard (IEEE 802.3 Clause 22) and extended (Clause 45) MDIO frame formats
- Supports legacy FEC buffer descriptors

53.2.1.2 IP protocol performance optimization features

- Operates on TCP/IP and UDP/IP and ICMP/IP protocol data or IP header only
- Enables wire-speed processing
- Supports IPv4 and IPv6
- Transparent passing of frames of other types and protocols
- Supports VLAN tagged frames according to IEEE 802.1q with transparent forwarding of VLAN tag and control field
- Automatic IP-header and payload (protocol specific) checksum calculation and verification on receive
- Automatic IP-header and payload (protocol specific) checksum generation and automatic insertion on transmit configurable on a per-frame basis
- Supports IP and TCP, UDP, ICMP data for checksum generation and checking
- Supports full header options for IPv4 and TCP protocol headers

- Provides IPv6 support to datagrams with base header only — datagrams with extension headers are passed transparently unmodifed/unchecked
- Provides statistics information for received IP and protocol errors
- Configurable automatic discard of erroneous frames
- Configurable automatic host-to-network (RX) and network-to-host (TX) byte order conversion for IP and TCP/UDP/ICMP headers within the frame
- Configurable padding remove for short IP datagrams on receive
- Configurable Ethernet payload alignment to allow for 32-bit word-aligned header and payload processing
- Programmable store-and-forward operation with clock and rate decoupling FIFOs

53.2.1.3 IEEE 1588 features

- Supports all IEEE 1588 frames.
- Allows reference clock to be chosen independently of network speed.
- Software-programmable precise time-stamping of ingress and egress frames
- Timer monitoring capabilities for system calibration and timing accuracy management
- Precise time-stamping of external events with programmable interrupt generation
- Programmable event and interrupt generation for external system control
- Supports hardware- and software-controllable timer synchronization.
- Provides a 4-channel IEEE 1588 timer. Each channel supports input capture and output compare using the 1588 counter.

53.2.2 Block diagram

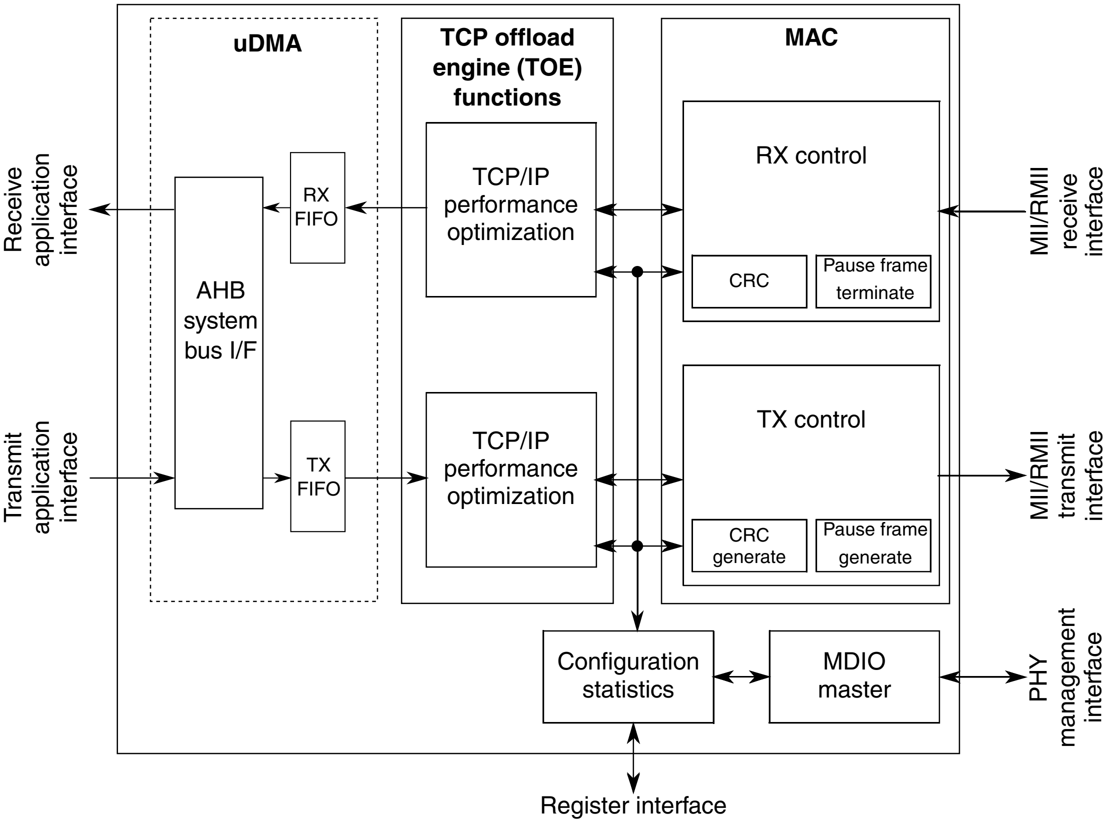

uDMA
TCP offload engine (TOE) functions
MAC
Receive application interface
RX FIFO
AHB system bus I/F
TCP/IP performance optimization
RX control
CRC
Pause frame terminate
MII/RMII receive interface
Transmit application interface
TX FIFO
TCP/IP performance optimization
TX control
CRC generate
Pause frame generate
MII/RMII transmit interface
Configuration statistics
MDIO master
PHY management interface
Register interface

Figure 53-1. Ethernet MAC-NET core block diagram

53.3 External signal description

NOTE

The MII column pertains only to devices that support MII.

MII	RMII	Description	I/O
MII_COL	—	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.	I
MII_CRS	—	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the RMII_CRS_DV pin.	I

Table continues on the next page...

MII	RMII	Description	I/O
MII_MDC	RMII_MDC	Output clock provides a timing reference to the PHY for data transfers on the MDIO signal.	O
MII_MDIO	RMII_MDIO	Transfers control information between the external PHY and the media-access controller. Data is synchronous to MDC. This signal is an input after reset.	I/O
MII_RXCLK	—	In MII mode, provides a timing reference for RXDV, RXD[3:0], and RXER.	I
MII_RXDV	RMII_CRS_DV	Asserting this input indicates the PHY has valid nibbles present on the MII. RXDV must remain asserted from the first recovered nibble of the frame through to the last nibble. Asserting RXDV must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal.	I
MII_RXD[3:0]	RMII_RXD[1:0]	Contains the Ethernet input data transferred from the PHY to the media-access controller when RXDV is asserted.	I
MII_RXER	RMII_RXER	When asserted with RXDV, indicates the PHY detects an error in the current frame.	I
MII_TXCLK	—	Input clock, which provides a timing reference for TXEN, TXD[3:0], and TXER.	I
MII_TXD[3:0]	RMII_TXD[1:0]	Serial output Ethernet data. Only valid during TXEN assertion.	O
MII_TXEN	RMII_TXEN	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is deasserted before the first TXCLK following the final nibble of the frame.	O
MII_TXER	—	When asserted for one or more clock cycles while TXEN is also asserted, PHY sends one or more illegal symbols.	O

Table continues on the next page...

MII	RMII	Description	I/O
—	RMII_REF_CLK	In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.	I
1588_TMR*n*	1588_TMR*n*	Capture/Compare block input/output event bus. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCR*n* register for inspection by software. When configured for compare, the corresponding signal 1588_TMR*n* is asserted for one cycle when the timer reaches the compare value programmed in ENET_TCCR*n*. An interrupt can be triggered if ENET_TCSR*n*[TIE] is set. A DMA request can be triggered if ENET_TCSR*n*[TDRE] is set.	I/O
ENET_1588_CLKIN	ENET_1588_CLKIN	Alternate IEEE 1588 Ethernet clock input; Clock period should be an integer number of nanoseconds	I

53.4 Memory map/register definition

ENET registers must be read or written with 32-bit accesses. Non-32 bit accesses will terminate with an error.

Reserved bits should be written with 0 and ignored on read. Unused registers read zero and a write has no effect.

This table shows Ethernet registers organization.

Table 53-1. Register map summary

Offset Address	Section	Description
0x0000 – 0x01FF	Configuration	Core control and status registers
0x0200 – 0x03FF	Statistics counters	MIB and Remote Network Monitoring (RFC 2819) registers
0x0400 – 0x0430	1588 control	1588 adjustable timer (TSM) and 1588 frame control
0x0600 – 0x07FC	Capture/Compare block	Registers for the Capture/Compare block

ENET memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Interrupt Event Register (ENET_EIR)	32	w1c	0000_0000h	53.4.1/2052
8	Interrupt Mask Register (ENET_EIMR)	32	R/W	0000_0000h	53.4.2/2055
10	Receive Descriptor Active Register (ENET_RDAR)	32	R/W	0000_0000h	53.4.3/2058
14	Transmit Descriptor Active Register (ENET_TDAR)	32	R/W	0000_0000h	53.4.4/2058
24	Ethernet Control Register (ENET_ECR)	32	R/W	F000_0000h	53.4.5/2059
40	MII Management Frame Register (ENET_MMFR)	32	R/W	0000_0000h	53.4.6/2061
44	MII Speed Control Register (ENET_MSCR)	32	R/W	0000_0000h	53.4.7/2062
64	MIB Control Register (ENET_MIBC)	32	R/W	C000_0000h	53.4.8/2064
84	Receive Control Register (ENET_RCR)	32	R/W	05EE_0001h	53.4.9/2065
C4	Transmit Control Register (ENET_TCR)	32	R/W	0000_0000h	53.4.10/ 2068
E4	Physical Address Lower Register (ENET_PALR)	32	R/W	0000_0000h	53.4.11/ 2070
E8	Physical Address Upper Register (ENET_PAUR)	32	R/W	0000_8808h	53.4.12/ 2070
EC	Opcode/Pause Duration Register (ENET_OPD)	32	R/W	0001_0000h	53.4.13/ 2071
118	Descriptor Individual Upper Address Register (ENET_IAUR)	32	R/W	0000_0000h	53.4.14/ 2071
11C	Descriptor Individual Lower Address Register (ENET_IALR)	32	R/W	0000_0000h	53.4.15/ 2072
120	Descriptor Group Upper Address Register (ENET_GAUR)	32	R/W	0000_0000h	53.4.16/ 2072
124	Descriptor Group Lower Address Register (ENET_GALR)	32	R/W	0000_0000h	53.4.17/ 2073
144	Transmit FIFO Watermark Register (ENET_TFWR)	32	R/W	0000_0000h	53.4.18/ 2073
180	Receive Descriptor Ring Start Register (ENET_RDSR)	32	R/W	0000_0000h	53.4.19/ 2074
184	Transmit Buffer Descriptor Ring Start Register (ENET_TDSR)	32	R/W	0000_0000h	53.4.20/ 2075
188	Maximum Receive Buffer Size Register (ENET_MRBR)	32	R/W	0000_0000h	53.4.21/ 2076
190	Receive FIFO Section Full Threshold (ENET_RSFL)	32	R/W	0000_0000h	53.4.22/ 2077
194	Receive FIFO Section Empty Threshold (ENET_RSEM)	32	R/W	0000_0000h	53.4.23/ 2077
198	Receive FIFO Almost Empty Threshold (ENET_RAEM)	32	R/W	0000_0004h	53.4.24/ 2078
19C	Receive FIFO Almost Full Threshold (ENET_RAFL)	32	R/W	0000_0004h	53.4.25/ 2078
1A0	Transmit FIFO Section Empty Threshold (ENET_TSEM)	32	R/W	0000_0000h	53.4.26/ 2079

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A4	Transmit FIFO Almost Empty Threshold (ENET_TAEM)	32	R/W	0000_0004h	53.4.27/ 2079
1A8	Transmit FIFO Almost Full Threshold (ENET_TAFL)	32	R/W	0000_0008h	53.4.28/ 2080
1AC	Transmit Inter-Packet Gap (ENET_TIPG)	32	R/W	0000_000Ch	53.4.29/ 2080
1B0	Frame Truncation Length (ENET_FTRL)	32	R/W	0000_07FFh	53.4.30/ 2081
1C0	Transmit Accelerator Function Configuration (ENET_TACC)	32	R/W	0000_0000h	53.4.31/ 2081
1C4	Receive Accelerator Function Configuration (ENET_RACC)	32	R/W	0000_0000h	53.4.32/ 2082
200	Reserved Statistic Register (ENET_RMON_T_DROP)	32	R	0000_0000h	53.4.33/ 2083
204	Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)	32	R	0000_0000h	53.4.34/ 2084
208	Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)	32	R	0000_0000h	53.4.35/ 2084
20C	Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)	32	R	0000_0000h	53.4.36/ 2085
210	Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)	32	R	0000_0000h	53.4.37/ 2085
214	Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)	32	R	0000_0000h	53.4.38/ 2086
218	Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)	32	R	0000_0000h	53.4.39/ 2086
21C	Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)	32	R	0000_0000h	53.4.40/ 2087
220	Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)	32	R	0000_0000h	53.4.41/ 2087
224	Tx Collision Count Statistic Register (ENET_RMON_T_COL)	32	R	0000_0000h	53.4.42/ 2088
228	Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)	32	R	0000_0000h	53.4.43/ 2088
22C	Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)	32	R	0000_0000h	53.4.44/ 2089
230	Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)	32	R	0000_0000h	53.4.45/ 2089
234	Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)	32	R	0000_0000h	53.4.46/ 2090
238	Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)	32	R	0000_0000h	53.4.47/ 2090
23C	Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)	32	R	0000_0000h	53.4.48/ 2091

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
240	Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)	32	R	0000_0000h	53.4.49/ 2091
244	Tx Octets Statistic Register (ENET_RMON_T_OCTETS)	32	R	0000_0000h	53.4.50/ 2092
248	IEEE_T_DROP Reserved Statistic Register (ENET_IEEE_T_DROP)	32	R	0000_0000h	53.4.51/ 2092
24C	Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)	32	R	0000_0000h	53.4.52/ 2092
250	Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)	32	R	0000_0000h	53.4.53/ 2093
254	Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)	32	R	0000_0000h	53.4.54/ 2093
258	Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)	32	R	0000_0000h	53.4.55/ 2094
25C	Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)	32	R	0000_0000h	53.4.56/ 2094
260	Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)	32	R	0000_0000h	53.4.57/ 2095
264	Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)	32	R	0000_0000h	53.4.58/ 2095
268	Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)	32	R	0000_0000h	53.4.59/ 2096
26C	ENET_IEEE_T_SQE	32	R (reads 0)	0000_0000h	53.4.60/ 2096
270	Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)	32	R	0000_0000h	53.4.61/ 2097
274	Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)	32	R	0000_0000h	53.4.62/ 2097
284	Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)	32	R	0000_0000h	53.4.63/ 2098
288	Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)	32	R	0000_0000h	53.4.64/ 2098
28C	Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)	32	R	0000_0000h	53.4.65/ 2099
290	Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)	32	R	0000_0000h	53.4.66/ 2099
294	Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)	32	R	0000_0000h	53.4.67/ 2100
298	Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE)	32	R	0000_0000h	53.4.68/ 2100
29C	Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)	32	R	0000_0000h	53.4.69/ 2101
2A0	Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)	32	R	0000_0000h	53.4.70/ 2101

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2A4	Reserved Statistic Register (ENET_RMON_R_RESVD_0)	32	R (reads 0)	0000_0000h	53.4.71/ 2101
2A8	Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)	32	R	0000_0000h	53.4.72/ 2102
2AC	Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)	32	R	0000_0000h	53.4.73/ 2102
2B0	Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)	32	R	0000_0000h	53.4.74/ 2103
2B4	Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)	32	R	0000_0000h	53.4.75/ 2103
2B8	Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)	32	R	0000_0000h	53.4.76/ 2104
2BC	Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)	32	R	0000_0000h	53.4.77/ 2104
2C0	Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)	32	R	0000_0000h	53.4.78/ 2105
2C4	Rx Octets Statistic Register (ENET_RMON_R_OCTETS)	32	R	0000_0000h	53.4.79/ 2105
2C8	Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)	32	R	0000_0000h	53.4.80/ 2106
2CC	Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)	32	R	0000_0000h	53.4.81/ 2106
2D0	Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)	32	R	0000_0000h	53.4.82/ 2107
2D4	Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)	32	R	0000_0000h	53.4.83/ 2107
2D8	Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)	32	R	0000_0000h	53.4.84/ 2108
2DC	Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)	32	R	0000_0000h	53.4.85/ 2108
2E0	Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)	32	R	0000_0000h	53.4.86/ 2109
400	Adjustable Timer Control Register (ENET_ATCR)	32	R/W	0000_0000h	53.4.87/ 2109
404	Timer Value Register (ENET_ATVR)	32	R/W	0000_0000h	53.4.88/ 2111
408	Timer Offset Register (ENET_ATOFF)	32	R/W	0000_0000h	53.4.89/ 2111
40C	Timer Period Register (ENET_ATPER)	32	R/W	3B9A_CA00h	53.4.90/ 2112
410	Timer Correction Register (ENET_ATCOR)	32	R/W	0000_0000h	53.4.91/ 2112
414	Time-Stamping Clock Period Register (ENET_ATINC)	32	R/W	0000_0000h	53.4.92/ 2113

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
418	Timestamp of Last Transmitted Frame (ENET_ATSTMP)	32	R	0000_0000h	53.4.93/ 2113
604	Timer Global Status Register (ENET_TGSR)	32	R/W	0000_0000h	53.4.94/ 2114
608	Timer Control Status Register (ENET_TCSR0)	32	R/W	0000_0000h	53.4.95/ 2115
60C	Timer Compare Capture Register (ENET_TCCR0)	32	R/W	0000_0000h	53.4.96/ 2116
610	Timer Control Status Register (ENET_TCSR1)	32	R/W	0000_0000h	53.4.95/ 2115
614	Timer Compare Capture Register (ENET_TCCR1)	32	R/W	0000_0000h	53.4.96/ 2116
618	Timer Control Status Register (ENET_TCSR2)	32	R/W	0000_0000h	53.4.95/ 2115
61C	Timer Compare Capture Register (ENET_TCCR2)	32	R/W	0000_0000h	53.4.96/ 2116
620	Timer Control Status Register (ENET_TCSR3)	32	R/W	0000_0000h	53.4.95/ 2115
624	Timer Compare Capture Register (ENET_TCCR3)	32	R/W	0000_0000h	53.4.96/ 2116

53.4.1 Interrupt Event Register (ENET_EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

NOTE

TxBD[INT] and RxBD[INT] must be set to 1 to allow setting the corresponding EIR register flags in enhanced mode, ENET_ECR[EN1588] = 1. Legacy mode does not require these flags to be enabled.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS_TIMER															
W	w1c	0		0	0			0	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_EIR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 BABR	Babbling Receive Error Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
2 BABT	Babbling Transmit Error Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused when a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
3 GRA	Graceful Stop Complete This interrupt is asserted after the transmitter is put into a pause state after completion of the frame currently being transmitted. See Graceful Transmit Stop (GTS) for conditions that lead to graceful stop. **NOTE:** The GRA interrupt is asserted only when the TX transitions into the stopped state. If this bit is cleared by writing 1 and the TX is still stopped, the bit is not set again.
4 TXF	Transmit Frame Interrupt Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
5 TXB	Transmit Buffer Interrupt Indicates a transmit buffer descriptor has been updated.
6 RXF	Receive Frame Interrupt Indicates a frame has been received and the last corresponding buffer descriptor has been updated.

Table continues on the next page...

ENET_EIR field descriptions (continued)

Field	Description
7 RXB	Receive Buffer Interrupt Indicates a receive buffer descriptor is not the last in the frame has been updated.
8 MII	MII Interrupt. Indicates that the MII has completed the data transfer requested.
9 EBERR	Ethernet Bus Error Indicates a system bus error occurred when a uDMA transaction is underway. When this bit is set, ECR[ETHEREN] is cleared, halting frame processing by the MAC. When this occurs, software must ensure proper actions, possibly resetting the system, to resume normal operation.
10 LC	Late Collision Indicates a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
11 RL	Collision Retry Limit Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half-duplex mode.
12 UN	Transmit FIFO Underrun Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13 PLR	Payload Receive Error Indicates a frame was received with a payload length error. See Frame Length/Type Verification: Payload Length Check for more information.
14 WAKEUP	Node Wakeup Request Indication Read-only status bit to indicate that a magic packet has been detected. Will act only if ECR[MAGICEN] is set.
15 TS_AVAIL	Transmit Timestamp Available Indicates that the timestamp of the last transmitted timing frame is available in the ATSTMP register.
16 TS_TIMER	Timestamp Timer The adjustable timer reached the period event. A period event interrupt can be generated if ATCR[PEREN] is set and the timer wraps according to the periodic setting in the ATPER register. Set the timer period value before setting ATCR[PEREN].
17–18 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
19 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
20–22 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24–31 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.

53.4.2 Interrupt Mask Register (ENET_EIMR)

EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR field (write 1 to clear) or a 0 is written to the EIMR field.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS_TIMER															
W		0		0	0			0	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_EIMR field descriptions

Field	Description
0 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
1 BABR	BABR Interrupt Mask Corresponds to interrupt source EIR[BABR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
2 BABT	BABT Interrupt Mask Corresponds to interrupt source EIR[BABT] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABT field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
3 GRA	GRA Interrupt Mask Corresponds to interrupt source EIR[GRA] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The

Table continues on the next page...

ENET_EIMR field descriptions (continued)

Field	Description
	corresponding EIR GRA field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
4 TXF	TXF Interrupt Mask Corresponds to interrupt source EIR[TXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
5 TXB	TXB Interrupt Mask Corresponds to interrupt source EIR[TXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
6 RXF	RXF Interrupt Mask Corresponds to interrupt source EIR[RXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
7 RXB	RXB Interrupt Mask Corresponds to interrupt source EIR[RXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXB field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
8 MII	MII Interrupt Mask Corresponds to interrupt source EIR[MII] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR MII field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
9 EBERR	EBERR Interrupt Mask Corresponds to interrupt source EIR[EBERR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR EBERR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
10 LC	LC Interrupt Mask Corresponds to interrupt source EIR[LC] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR LC field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

Table continues on the next page...

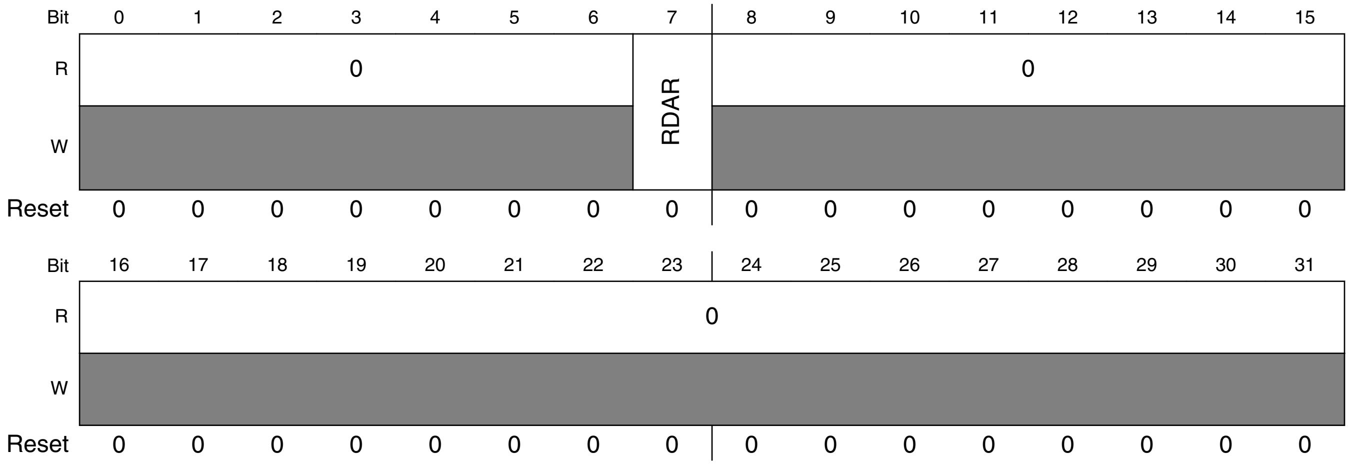
ENET_EIMR field descriptions (continued)

Field	Description
11 RL	RL Interrupt Mask Corresponds to interrupt source EIR[RL] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
12 UN	UN Interrupt Mask Corresponds to interrupt source EIR[UN] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR UN field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
13 PLR	PLR Interrupt Mask Corresponds to interrupt source EIR[PLR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR PLR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
14 WAKEUP	WAKEUP Interrupt Mask Corresponds to interrupt source EIR[WAKEUP] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR WAKEUP field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
15 TS_AVAIL	TS_AVAIL Interrupt Mask Corresponds to interrupt source EIR[TS_AVAIL] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_AVAIL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
16 TS_TIMER	TS_TIMER Interrupt Mask Corresponds to interrupt source EIR[TS_TIMER] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_TIMER field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
17–18 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
19 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
20–22 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24–31 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.

53.4.3 Receive Descriptor Active Register (ENET_RDAR)

RDAR is a command register, written by the user, to indicate that the receive descriptor ring has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: 0h base + 10h offset = 10h



ENET_RDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.4 Transmit Descriptor Active Register (ENET_TDAR)

The TDAR is a command register that the user writes to indicate that the transmit descriptor ring has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

The TDAR register is cleared at reset, when ECR[ETHEREN] transitions from set to cleared, or when ECR[RESET] is set.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							TDAR	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.5 Ethernet Control Register (ENET_ECR)

ECR is a read/write user register, though hardware may also alter fields in this register. It controls many of the high level features of the Ethernet MAC, including legacy FEC support through the EN1588 field.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved				0	0	0	DBSWP	STOPEN	DBGEN		EN1588	SLEEP	MAGICEN	ETHEREN	RESET
W											0					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ECR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This field must be set to F_0000h.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 DBSWP	Descriptor Byte Swapping Enable Swaps the byte locations of the buffer descriptors. **NOTE:** This field resets to 0 and must not be changed. 0 The buffer descriptor bytes are not swapped to support big-endian devices. 1 The buffer descriptor bytes are swapped to support little-endian devices.
24 STOPEN	STOPEN Signal Control Controls device behavior in doze mode. In doze mode, if this field is set then all the clocks of the ENET assembly are disabled, except the RMII /MII clock. Doze mode is similar to a conditional stop mode entry for the ENET assembly depending on ECR[STOPEN]. **NOTE:** If module clocks are gated in this mode, the module can still wake the system after receiving a magic packet in stop mode. MAGICEN must be set prior to entering sleep/stop mode.
25 DBGEN	Debug Enable Enables the MAC to enter hardware freeze mode when the device enters debug mode. 0 MAC continues operation in debug mode. 1 MAC enters hardware freeze mode when the processor is in debug mode.
26 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
27 EN1588	EN1588 Enable Enables enhanced functionality of the MAC. 0 Legacy FEC buffer descriptors and functions enabled. 1 Enhanced frame time-stamping functions enabled.
28 SLEEP	Sleep Mode Enable 0 Normal operating mode. 1 Sleep mode.
29 MAGICEN	Magic Packet Detection Enable Enables/disables magic packet detection. **NOTE:** MAGICEN is relevant only if the SLEEP field is set. If MAGICEN is set, changing the SLEEP field enables/disables sleep mode and magic packet detection. 0 Magic detection logic disabled. 1 The MAC core detects magic packets and asserts EIR[WAKEUP] when a frame is detected.

Table continues on the next page...

ENET_ECR field descriptions (continued)

Field	Description
30 ETHEREN	Ethernet Enable Enables/disables the Ethernet MAC. When the MAC is disabled, the buffer descriptors for an aborted transmit frame are not updated. The uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware clears this field under the following conditions: • RESET is set by software • An error condition causes the EBERR field to set. **NOTE:** • ETHEREN must be set at the very last step during ENET configuration/setup/initialization, only *after* all other ENET-related registers have been configured. • If ETHEREN is cleared to 0 by software then next time ETHEREN is set, the EIR interrupts must cleared to 0 due to previous pending interrupts. 0 Reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. 1 MAC is enabled, and reception and transmission are possible.
31 RESET	Ethernet MAC Reset When this field is set, it clears the ETHEREN field.

53.4.6 MII Management Frame Register (ENET_MMFR)

Writing to MMFR triggers a management frame transaction to the PHY device unless MSCR is programmed to zero.

If MSCR is changed from zero to non-zero during a write to MMFR, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the EIR[MII] interrupt indication to avoid writing to the MMFR register while frame generation is in progress.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST		OP		PA					RA					TA		DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MMFR field descriptions

Field	Description
0–1 ST	Start Of Frame Delimiter See Table 53-39 (Clause 22) or Table 53-41 (Clause 45) for correct value.

Table continues on the next page...

ENET_MMFR field descriptions (continued)

Field	Description
2–3 OP	Operation Code See Table 53-39 (Clause 22) or Table 53-41 (Clause 45) for correct value.
4–8 PA	PHY Address See Table 53-39 (Clause 22) or Table 53-41 (Clause 45) for correct value.
9–13 RA	Register Address See Table 53-39 (Clause 22) or Table 53-41 (Clause 45) for correct value.
14–15 TA	Turn Around This field must be programmed to 10 to generate a valid MII management frame.
16–31 DATA	Management Frame Data This is the field for data to be written to or read from the PHY register.

53.4.7 MII Speed Control Register (ENET_MSCR)

MSCR provides control of the MII clock (MDC pin) frequency and allows a preamble drop on the MII management frame.

The MII_SPEED field must be programmed with a value to provide an MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be cleared to turn off MDC. The MDC signal generated has a 50% duty cycle except when MII_SPEED changes during operation. This change takes effect following a rising or falling edge of MDC.

For example, if the internal module clock (i.e., IPS bus clock) is 25 MHz, programming MII_SPEED to 0x4 results in an MDC as given in the following equation:

MII clock frequency = 25 MHz / ((4 + 1) x 2) = 2.5 MHz

The following table shows the optimum values for MII_SPEED as a function of IPS bus clock frequency.

Table 53-2. Programming Examples for MSCR

Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
25 MHz	0x4	2.50 MHz
33 MHz	0x6	2.36 MHz
40 MHz	0x7	2.50 MHz
50 MHz	0x9	2.50 MHz

Table continues on the next page...

Table 53-2. Programming Examples for MSCR (continued)

Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
66 MHz	0xD	2.36 MHz

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					HOLDTIME			DIS_PRE	MII_SPEED						0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MSCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 HOLDTIME	Hold time On MDIO Output IEEE802.3 clause 22 defines a minimum of 10 ns for the hold time on the MDIO output. Depending on the host bus frequency, the setting may need to be increased. 000 1 internal module clock cycle 001 2 internal module clock cycles 010 3 internal module clock cycles 111 8 internal module clock cycles
24 DIS_PRE	Disable Preamble Enables/disables prepending a preamble to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY devices do not require it. 0 Preamble enabled. 1 Preamble (32 ones) is not prepended to the MII management frame.
25–30 MII_SPEED	MII Speed Controls the frequency of the MII management interface clock (MDC) relative to the internal module clock. A value of 0 in this field turns off MDC and leaves it in low voltage state. Any non-zero value results in the MDC frequency of: 1/((MII_SPEED + 1) x 2) of the internal module clock frequency
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.8 MIB Control Register (ENET_MIBC)

MIBC is a read/write register controlling and observing the state of the MIB block. Access this register to disable the MIB block operation or clear the MIB counters. The MIB_DIS field resets to 1.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIB_DIS	MIB_IDLE	MIB_CLEAR	0												
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MIBC field descriptions

Field	Description
0 MIB_DIS	Disable MIB Logic If this control field is set, 0 MIB logic is enabled. 1 MIB logic is disabled. The MIB logic halts and does not update any MIB counters.

Table continues on the next page...

ENET_MIBC field descriptions (continued)

Field	Description
1 MIB_IDLE	MIB Idle 0 The MIB block is updating MIB counters. 1 The MIB block is not currently updating any MIB counters.
2 MIB_CLEAR	MIB Clear **NOTE:** This field is not self-clearing. To clear the MIB counters set and then clear this field. 0 See note above. 1 All statistics counters are reset to 0.
3–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.9 Receive Control Register (ENET_RCR)

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GRS	NLC	MAX_FL													
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	0	1	1	1	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFEN	CRCFWD	PAUFWD	PADEN			RMII_10T	RMII_MODE			FCE	BC_REJ	PROM	MII_MODE	DRT	LOOP
W					0				0	0						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

ENET_RCR field descriptions

Field	Description
0 GRS	Graceful Receive Stopped Read-only status indicating that the MAC receive datapath is stopped.
1 NLC	Payload Length Check Disable Enables/disables a payload length check. 0 The payload length check is disabled. 1 The core checks the frame's payload length with the frame length/type field. Errors are indicated in the EIR[PLC] field.
2–15 MAX_FL	Maximum Frame Length Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL cause the BABT interrupt to occur. Receive frames longer than MAX_FL cause the BABR interrupt to occur and set the LG field in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
16 CFEN	MAC Control Frame Enable Enables/disables the MAC control frame. 0 MAC control frames with any opcode other than 0x0001 (pause frame) are accepted and forwarded to the client interface. 1 MAC control frames with any opcode other than 0x0001 (pause frame) are silently discarded.
17 CRCFWD	Terminate/Forward Received CRC Specifies whether the CRC field of received frames is transmitted or stripped. **NOTE:** If padding function is enabled (PADEN = 1), CRCFWD is ignored and the CRC field is checked and always terminated and removed. 0 The CRC field of received frames is transmitted to the user application. 1 The CRC field is stripped from the frame.
18 PAUFWD	Terminate/Forward Pause Frames Specifies whether pause frames are terminated or forwarded. 0 Pause frames are terminated and discarded in the MAC. 1 Pause frames are forwarded to the user application.
19 PADEN	Enable Frame Padding Remove On Receive Specifies whether the MAC removes padding from received frames. 0 No padding is removed on receive by the MAC. 1 Padding is removed from received frames.
20–21 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
22 RMII_10T	Enables 10-Mbit/s mode of the RMII . 0 100-Mbit/s operation. 1 10-Mbit/s operation.

Table continues on the next page...

ENET_RCR field descriptions (continued)

Field	Description
23 RMII_MODE	RMII Mode Enable Specifies whether the MAC is configured for MII mode or RMII operation . 0 MAC configured for MII mode. 1 MAC configured for RMII operation.
24 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
25 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
26 FCE	Flow Control Enable If set, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
27 BC_REJ	Broadcast Frame Reject If set, frames with destination address (DA) equal to 0xFFFF_FFFF_FFFF are rejected unless the PROM field is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the MISS (M) is set in the receive buffer descriptor.
28 PROM	Promiscuous Mode All frames are accepted regardless of address matching. 0 Disabled. 1 Enabled.
29 MII_MODE	Media Independent Interface Mode This field must always be set. 0 Reserved. 1 MII or RMII mode, as indicated by the RMII_MODE field.
30 DRT	Disable Receive On Transmit 0 Receive path operates independently of transmit. Used for full-duplex or to monitor transmit activity in half-duplex mode. 1 Disable reception of frames while transmitting. Normally used for half-duplex mode.
31 LOOP	Internal Loopback This is an MII internal loopback, therefore MII_MODE must be written to 1 and RMII_MODE must be written to 0. 0 Loopback disabled. 1 Transmitted frames are looped back internal to the device and transmit MII output signals are not asserted. DRT must be cleared.

53.4.10 Transmit Control Register (ENET_TCR)

TCR is read/write and configures the transmit block. This register is cleared at system reset. FDEN can only be modified when ECR[ETHEREN] is cleared.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					Reserved	CRCFWD	ADDINS	ADDSEL			RFC_PAUSE	TFC_PAUSE	FDEN		GTS
W															0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This field is read/write and must be set to 0.
22 CRCFWD	Forward Frame From Application With CRC

Table continues on the next page...

ENET_TCR field descriptions (continued)

Field	Description
	0 TxBD[TC] controls whether the frame has a CRC from the application. 1 The transmitter does not append any CRC to transmitted frames, as it is expecting a frame with CRC from the application.
23 ADDINS	Set MAC Address On Transmit 0 The source MAC address is not modified by the MAC. 1 The MAC overwrites the source MAC address with the programmed MAC address according to ADDSEL.
24–26 ADDSEL	Source MAC Address Select On Transmit If ADDINS is set, indicates the MAC address that overwrites the source MAC address. 000 Node MAC address programmed on PADDR1/2 registers. 100 Reserved. 101 Reserved. 110 Reserved.
27 RFC_PAUSE	Receive Frame Control Pause This status field is set when a full-duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This field automatically clears when the pause duration is complete.
28 TFC_PAUSE	Transmit Frame Control Pause Pauses frame transmission. When this field is set, EIR[GRA] is set. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC control PAUSE frame. 0 No PAUSE frame transmitted. 1 The MAC stops transmission of data frames after the current transmission is complete.
29 FDEN	Full-Duplex Enable If this field is set, frames transmit independent of carrier sense and collision inputs. Only modify this bit when ECR[ETHEREN] is cleared.
30 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
31 GTS	Graceful Transmit Stop When this field is set, MAC stops transmission after any frame currently transmitted is complete and EIR[GRA] is set. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear ECR[ETHEREN] following the GRA interrupt.

53.4.11 Physical Address Lower Register (ENET_PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the six-byte source address field when transmitting PAUSE frames.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_PALR field descriptions

Field	Description
0–31 PADDR1	Pause Address Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

53.4.12 Physical Address Upper Register (ENET_PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the six-byte source address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (0x8808) for transmission of PAUSE frames.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PADDR2																TYPE															
W																	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

ENET_PAUR field descriptions

Field	Description
0–15 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
16–31 TYPE	Type Field In PAUSE Frames These fields have a constant value of 0x8808.

53.4.13 Opcode/Pause Duration Register (ENET_OPD)

OPD is read/write accessible. This register contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0x0001. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field. The lower 16 bits of this register are not reset and you must initialize it.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OPCODE																PAUSE_DUR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_OPD field descriptions

Field	Description
0–15 OPCODE	Opcode Field In PAUSE Frames These fields have a constant value of 0x0001.
16–31 PAUSE_DUR	Pause Duration Pause duration field used in PAUSE frames.

53.4.14 Descriptor Individual Upper Address Register (ENET_IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IAUR field descriptions

Field	Description
0–31 IADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

53.4.15 Descriptor Individual Lower Address Register (ENET_IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: 0h base + 11Ch offset = 11Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IALR field descriptions

Field	Description
0–31 IADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

53.4.16 Descriptor Group Upper Address Register (ENET_GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_GAUR field descriptions

Field	Description
0–31 GADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

53.4.17 Descriptor Group Lower Address Register (ENET_GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_GALR field descriptions

Field	Description
0–31 GADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

53.4.18 Transmit FIFO Watermark Register (ENET_TFWR)

If TFWR[STRFWD] is cleared, TFWR[TFWR] controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement, for example, worst-case bus access latency by the transmit data uDMA channel.

When the FIFO level reaches the value the TFWR field and when the STR_FWD is set to ‘0’, the MAC transmit control logic starts frame transmission even before the end-of-frame is available in the FIFO (cut-through operation).

If a complete frame has a size smaller than the threshold programmed with TFWR, the MAC also transmits the Frame to the line.

To enable store and forward on the Transmit path, set STR_FWD to ‘1’. In this case, the MAC starts to transmit data only when a complete frame is stored in the Transmit FIFO.

Address: 0h base + 144h offset = 144h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							STRFWD	0		TFWR					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TFWR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 STRFWD	Store And Forward Enable 0 Reset. The transmission start threshold is programmed in TFWR[TFWR]. 1 Enabled.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 TFWR	Transmit FIFO Write If TFWR[STRFWD] is cleared, this field indicates the number of bytes, in steps of 64 bytes, written to the transmit FIFO before transmission of a frame begins. **NOTE:** If a frame with less than the threshold is written, it is still sent independently of this threshold setting. The threshold is relevant only if the frame is larger than the threshold given. 000000 64 bytes written. 000001 64 bytes written. 000010 128 bytes written. 000011 192 bytes written. ... ... 011111 1984 bytes written.

53.4.19 Receive Descriptor Ring Start Register (ENET_RDSR)

RDSR points to the beginning of the circular receive buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, it is recommended to be 128-bit aligned, that is, evenly divisible by 16.

NOTE

This register must be initialized prior to operation

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RDSR field descriptions

Field	Description
0–28 R_DES_START	Pointer to the beginning of the receive buffer descriptor queue.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.20 Transmit Buffer Descriptor Ring Start Register (ENET_TDSR)

TDSR provides a pointer to the beginning of the circular transmit buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, it is recommended to be 128-bit aligned, that is, evenly divisible by 16.

NOTE

This register must be initialized prior to operation.

Address: 0h base + 184h offset = 184h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	X_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDSR field descriptions

Field	Description
0–28 X_DES_START	Pointer to the beginning of the transmit buffer descriptor queue.

Table continues on the next page...

ENET_TDSR field descriptions (continued)

Field	Description
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.21 Maximum Receive Buffer Size Register (ENET_MRBR)

The MRBR is a user-programmable register that dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To allow one maximum size frame per buffer, MRBR must be set to RCR[MAX_FL] or larger.
- To properly align the buffer, MRBR must be evenly divisible by 16. To ensure this, the lower four bits are set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR greater than or equal to 256 bytes.

NOTE

This register must be initialized before operation.

Address: 0h base + 188h offset = 188h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																					R_BUF_SIZE							0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MRBR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–27 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
28–31 Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

53.4.22 Receive FIFO Section Full Threshold (ENET_RSFL)

Address: 0h base + 190h offset = 190h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								RX_SECTION_FULL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RSFL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RX_SECTION_ FULL	Value Of Receive FIFO Section Full Threshold Value, in 64-bit words, of the receive FIFO section full threshold. Clear this field to enable store and forward on the RX FIFO. When programming a value greater than 0 (cut-through operation), it must be greater than RAEM[RX_ALMOST_EMPTY]. When the FIFO level reaches the value in this field, data is available in the Receive FIFO (cut-through operation).

53.4.23 Receive FIFO Section Empty Threshold (ENET_RSEM)

Address: 0h base + 194h offset = 194h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											STAT_ SECTION_ EMPTY					0								RX_SECTION_EMPTY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RSEM field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 STAT_ SECTION_ EMPTY	RX Status FIFO Section Empty Threshold Defines number of frames in the receive FIFO, independent of its size, that can be accepted. If the limit is reached, reception will continue normally, however a pause frame will be triggered to indicate a possible congestion to the remote device to avoid FIFO overflow. A value of 0 disables automatic pause frame generation
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RX_SECTION_ EMPTY	Value Of The Receive FIFO Section Empty Threshold Value, in 64-bit words, of the receive FIFO section empty threshold. When the FIFO has reached this level, a pause frame will be issued.

Table continues on the next page...

ENET_RSEM field descriptions (continued)

Field	Description
	A value of 0 disables automatic pause frame generation. When the FIFO level goes below the value programmed in this field, an XON pause frame is issued to indicate the FIFO congestion is cleared to the remote Ethernet client. **NOTE:** The section-empty threshold indications from both FIFOs are OR'ed to cause XOFF pause frame generation.

53.4.24 Receive FIFO Almost Empty Threshold (ENET_RAEM)

Address: 0h base + 198h offset = 198h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								RX_ALMOST_EMPTY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

ENET_RAEM field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RX_ALMOST_EMPTY	Value Of The Receive FIFO Almost Empty Threshold Value, in 64-bit words, of the receive FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field and the end-of-frame has not been received for the frame yet, the core receive read control stops FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. A minimum value of 4 should be set.

53.4.25 Receive FIFO Almost Full Threshold (ENET_RAFL)

Address: 0h base + 19Ch offset = 19Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								RX_ALMOST_FULL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

ENET_RAFL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ENET_RAFL field descriptions (continued)

Field	Description
24–31 RX_ALMOST_ FULL	Value Of The Receive FIFO Almost Full Threshold Value, in 64-bit words, of the receive FIFO almost full threshold. When the FIFO level comes close to the maximum, so that there is no more space for at least RX_ALMOST_FULL number of words, the MAC stops writing data in the FIFO and truncates the received frame to avoid FIFO overflow. The corresponding error status will be set when the frame is delivered to the application. A minimum value of 4 should be set.

53.4.26 Transmit FIFO Section Empty Threshold (ENET_TSEM)

Address: 0h base + 1A0h offset = 1A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								TX_SECTION_EMPTY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TSEM field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TX_SECTION_ EMPTY	Value Of The Transmit FIFO Section Empty Threshold Value, in 64-bit words, of the transmit FIFO section empty threshold. See Transmit FIFO for more information.

53.4.27 Transmit FIFO Almost Empty Threshold (ENET_TAEM)

Address: 0h base + 1A4h offset = 1A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								TX_ALMOST_EMPTY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

ENET_TAEM field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TX_ALMOST_ EMPTY	Value of Transmit FIFO Almost Empty Threshold Value, in 64-bit words, of the transmit FIFO almost empty threshold.

Table continues on the next page...

ENET_TAEM field descriptions (continued)

Field	Description
	When the FIFO level reaches the value programmed in this field, and no end-of-frame is available for the frame, the MAC transmit logic, to avoid FIFO underflow, stops reading the FIFO and transmits a frame with an MII error indication. See Transmit FIFO for more information. A minimum value of 4 should be set.

53.4.28 Transmit FIFO Almost Full Threshold (ENET_TAFL)

Address: 0h base + 1A8h offset = 1A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								TX_ALMOST_FULL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

ENET_TAFL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TX_ALMOST_FULL	Value Of The Transmit FIFO Almost Full Threshold Value, in 64-bit words, of the transmit FIFO almost full threshold. A minimum value of six is required . A recommended value of at least 8 should be set allowing a latency of two clock cycles to the application. If more latency is required the value can be increased as necessary (latency = TAFL - 5). When the FIFO level comes close to the maximum, so that there is no more space for at least TX_ALMOST_FULL number of words, the pin ff_tx_rdy is deasserted. If the application does not react on this signal, the FIFO write control logic, to avoid FIFO overflow, truncates the current frame and sets the error status. As a result, the frame will be transmitted with an GMII/MII error indication. See Transmit FIFO for more information. **NOTE:** A FIFO overflow is a fatal error and requires a global reset on the transmit datapath or at least deassertion of ETHEREN.

53.4.29 Transmit Inter-Packet Gap (ENET_TIPG)

Address: 0h base + 1ACh offset = 1ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											IPG				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

ENET_TIPG field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IPG	Transmit Inter-Packet Gap Indicates the IPG, in bytes, between transmitted frames. Valid values range from 8 to 26. If the written value is less than 8 or greater than 26, the internal (effective) IPG is 12. **NOTE:** The IPG value read will be the value that was written, even if it is out of range.

53.4.30 Frame Truncation Length (ENET_FTRL)

Address: 0h base + 1B0h offset = 1B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																		TRUNC_FL													
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

ENET_FTRL field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 TRUNC_FL	Frame Truncation Length Indicates the value a receive frame is truncated, if it is greater than this value. Must be greater than or equal to RCR[MAX_FL]. **NOTE:** Truncation happens at TRUNC_FL. However, when truncation occurs, the application (FIFO) may receive less data, guaranteeing that it never receives more than the set limit.

53.4.31 Transmit Accelerator Function Configuration (ENET_TACC)

TACC controls accelerator actions when sending frames. The register can be changed before or after each frame, but it must remain unmodified during frame writes into the transmit FIFO.

The TFWR[STRFWD] field must be set to use the checksum feature.

Address: 0h base + 1C0h offset = 1C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R												PROCHK	IPCHK			SHIFT16
W	0													0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TACC field descriptions

Field	Description
0–26 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
27 PROCHK	Enables insertion of protocol checksum. 0 Checksum not inserted. 1 If an IP frame with a known protocol is transmitted, the checksum is inserted automatically into the frame. The checksum field must be cleared. The other frames are not modified.
28 IPCHK	Enables insertion of IP header checksum. 0 Checksum is not inserted. 1 If an IP frame is transmitted, the checksum is inserted automatically. The IP header checksum field must be cleared. If a non-IP frame is transmitted the frame is not modified.
29–30 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
31 SHIFT16	TX FIFO Shift-16 0 Disabled. 1 Indicates to the transmit data FIFO that the written frames contain two additional octets before the frame data. This means the actual frame begins at bit 16 of the first word written into the FIFO. This function allows putting the frame payload on a 32-bit boundary in memory, as the 14-byte Ethernet header is extended to a 16-byte header.

53.4.32 Receive Accelerator Function Configuration (ENET_RACC)

Address: 0h base + 1C4h offset = 1C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									SHIFT16	LINEDIS				PRODIS	IPDIS	PADREM
W	0										0					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RACC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24 SHIFT16	RX FIFO Shift-16 When this field is set, the actual frame data starts at bit 16 of the first word read from the RX FIFO aligning the Ethernet payload on a 32-bit boundary. **NOTE:** This function only affects the FIFO storage and has no influence on the statistics, which use the actual length of the frame received. 0 Disabled. 1 Instructs the MAC to write two additional bytes in front of each frame received into the RX FIFO.
25 LINEDIS	Enable Discard Of Frames With MAC Layer Errors 0 Frames with errors are not discarded. 1 Any frame received with a CRC, length, or PHY error is automatically discarded and not forwarded to the user application interface.
26–28 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
29 PRODIS	Enable Discard Of Frames With Wrong Protocol Checksum 0 Frames with wrong checksum are not discarded. 1 If a TCP/IP, UDP/IP, or ICMP/IP frame is received that has a wrong TCP, UDP, or ICMP checksum, the frame is discarded. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
30 IPDIS	Enable Discard Of Frames With Wrong IPv4 Header Checksum 0 Frames with wrong IPv4 header checksum are not discarded. 1 If an IPv4 frame is received with a mismatching header checksum, the frame is discarded. IPv6 has no header checksum and is not affected by this setting. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
31 PADREM	Enable Padding Removal For Short IP Frames 0 Padding not removed. 1 Any bytes following the IP payload section of the frame are removed from the frame.

53.4.33 Reserved Statistic Register (ENET_RMON_T_DROP)

Address: 0h base + 200h offset = 200h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_DROP field descriptions

Field	Description
0–31 Reserved	This read-only field always has the value 0. This field is reserved.

53.4.34 Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)

Address: 0h base + 204h offset = 204h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_PACKETS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Packet count Transmit packet count

53.4.35 Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)

RMON Tx Broadcast Packets

Address: 0h base + 208h offset = 208h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_BC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Broadcast packets

53.4.36 Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)

Address: 0h base + 20Ch offset = 20Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_MC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Multicast packets

53.4.37 Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_CRC_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Packets with CRC/align error

53.4.38 Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_UNDERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets less than 64 bytes with good CRC

53.4.39 Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)

Address: 0h base + 218h offset = 218h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_OVERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than MAX_FL bytes with good CRC

53.4.40 Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)

.

Address: 0h base + 21Ch offset = 21Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_FRAG field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of packets less than 64 bytes with bad CRC

53.4.41 Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_JAB field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than MAX_FL bytes and bad CRC

53.4.42 Tx Collision Count Statistic Register (ENET_RMON_T_COL)

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_COL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit collisions

53.4.43 Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)

.

Address: 0h base + 228h offset = 228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P64 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 64-byte transmit packets

53.4.44 Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)

Address: 0h base + 22Ch offset = 22Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P65TO127 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 65- to 127-byte transmit packets

53.4.45 Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)

Address: 0h base + 230h offset = 230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P128TO255 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 128- to 255-byte transmit packets

53.4.46 Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P256TO511 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 256- to 511-byte transmit packets

53.4.47 Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)

.

Address: 0h base + 238h offset = 238h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P512TO1023 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 512- to 1023-byte transmit packets

53.4.48 Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)

Address: 0h base + 23Ch offset = 23Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P1024TO2047 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 1024- to 2047-byte transmit packets

53.4.49 Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P_GTE2048 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than 2048 bytes

53.4.50 Tx Octets Statistic Register (ENET_RMON_T_OCTETS)

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXOCTS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_OCTETS field descriptions

Field	Description
0–31 TXOCTS	Number of transmit octets

53.4.51 IEEE_T_DROP Reserved Statistic Register (ENET_IEEE_T_DROP)

Address: 0h base + 248h offset = 248h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_DROP field descriptions

Field	Description
0–31 Reserved	This read-only field always has the value 0. This field is reserved.

53.4.52 Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)

Address: 0h base + 24Ch offset = 24Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_FRAME_OK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted OK **NOTE:** Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

53.4.53 Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)

Address: 0h base + 250h offset = 250h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_1COL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with one collision

53.4.54 Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)

Address: 0h base + 254h offset = 254h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_MCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with multiple collisions

53.4.55 Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)

Address: 0h base + 258h offset = 258h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_DEF field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with deferral delay

53.4.56 Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)

Address: 0h base + 25Ch offset = 25Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_LCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with late collision

53.4.57 Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)

Address: 0h base + 260h offset = 260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_EXCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with excessive collisions

53.4.58 Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)

Address: 0h base + 264h offset = 264h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_MACERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with transmit FIFO underrun

53.4.59 Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)

Address: 0h base + 268h offset = 268h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_CSERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with carrier sense error

53.4.60 ENET_IEEE_T_SQE

Address: 0h base + 26Ch offset = 26Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_SQE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with SQE error **NOTE:** Counter not implemented (always reads zero) as no SQE information is available.

53.4.61 Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)

Address: 0h base + 270h offset = 270h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_FDXFC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of flow-control pause frames transmitted

53.4.62 Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)

Address: 0h base + 274h offset = 274h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_OCTETS_OK field descriptions

Field	Description
0–31 COUNT	Octet count for frames transmitted without error **NOTE** Counts total octets (includes header and FCS fields).

53.4.63 Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)

Address: 0h base + 284h offset = 284h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_PACKETS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of packets received

53.4.64 Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)

Address: 0h base + 288h offset = 288h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_BC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive broadcast packets

53.4.65 Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)

Address: 0h base + 28Ch offset = 28Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_MC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive multicast packets

53.4.66 Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)

Address: 0h base + 290h offset = 290h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_CRC_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with CRC or align error

53.4.67 Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)

Address: 0h base + 294h offset = 294h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_UNDERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with less than 64 bytes and good CRC

53.4.68 Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE)

Address: 0h base + 298h offset = 298h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_OVERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets greater than MAX_FL and good CRC

53.4.69 Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)

Address: 0h base + 29Ch offset = 29Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_FRAG field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with less than 64 bytes and bad CRC

53.4.70 Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)

Address: 0h base + 2A0h offset = 2A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_JAB field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets greater than MAX_FL and bad CRC

53.4.71 Reserved Statistic Register (ENET_RMON_R_RESVD_0)

Address: 0h base + 2A4h offset = 2A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_RESVD_0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.72 Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)

Address: 0h base + 2A8h offset = 2A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P64 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 64-byte receive packets

53.4.73 Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)

Address: 0h base + 2ACh offset = 2ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P65TO127 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 65- to 127-byte recieve packets

53.4.74 Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P128TO255 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 128- to 255-byte recieve packets

53.4.75 Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P256TO511 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 256- to 511-byte recieve packets

53.4.76 Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)

Address: 0h base + 2B8h offset = 2B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P512TO1023 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 512- to 1023-byte recieve packets

53.4.77 Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)

Address: 0h base + 2BCh offset = 2BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P1024TO2047 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 1024- to 2047-byte recieve packets

53.4.78 Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)

Address: 0h base + 2C0h offset = 2C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P_GTE2048 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of greater-than-2048-byte recieve packets

53.4.79 Rx Octets Statistic Register (ENET_RMON_R_OCTETS)

Address: 0h base + 2C4h offset = 2C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_OCTETS field descriptions

Field	Description
0–31 COUNT	Number of receive octets

53.4.80 Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)

Counter increments if a frame with invalid or missing SFD character is detected and has been dropped. None of the other counters increments if this counter increments.

Address: 0h base + 2C8h offset = 2C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_DROP field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Frame count

53.4.81 Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)

Address: 0h base + 2CCh offset = 2CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_FRAME_OK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received OK

53.4.82 Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)

Address: 0h base + 2D0h offset = 2D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_CRC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received with CRC error

53.4.83 Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)

Address: 0h base + 2D4h offset = 2D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received with alignment error

53.4.84 Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)

Address: 0h base + 2D8h offset = 2D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_MACERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Receive FIFO overflow count

53.4.85 Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)

Address: 0h base + 2DCh offset = 2DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_FDXFC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of flow-control pause frames received

53.4.86 Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_OCTETS_OK field descriptions

Field	Description
0–31 COUNT	Number of octets for frames received without error **NOTE:** Counts total octets (includes header and FCS fields). Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

53.4.87 Adjustable Timer Control Register (ENET_ATCR)

ATCR command fields can trigger the corresponding events directly. It is not necessary to preserve any of the configuration fields when a command field is set in the register, that is, no read-modify-write is required.

Address: 0h base + 400h offset = 400h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		SLAVE	0	CAPTURE	0	RESTART		PINPER			PEREN	OFFRST	OFFEN		EN
W								0		0	1				0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATCR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SLAVE	Enable Timer Slave Mode 0 The timer is active and all configuration fields in this register are relevant. 1 The internal timer is disabled and the externally provided timer value is used. All other fields, except CAPTURE, in this register have no effect. CAPTURE can still be used to capture the current timer value.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 CAPTURE	Capture Timer Value When this field is set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. **NOTE:** To ensure that the correct time value is read from the ATVR register, a minimum amount of time must elapse from issuing this command to reading the ATVR register. This minimum time is defined by the greater of either six register clock cycles or six 1588/timestamp clock cycles. 0 No effect. 1 The current time is captured and can be read from the ATVR register.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RESTART	Reset Timer Resets the timer to zero. This has no effect on the counter enable. If the counter is enabled when this field is set, the timer is reset to zero and starts counting from there. When set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. **NOTE:** The Reset Timer command requires at least 6 clock cycles of either the register clock or the 1588/timestamp clock, whichever is greater, to complete.
23 Reserved	This field is reserved.
24 PINPER	Enables event signal output assertion on period event. **NOTE:** Not all devices contain the event signal output. See the chip configuration details. 0 Disable. 1 Enable.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved. **NOTE:** This field must be written always with one.
27 PEREN	Enable Periodical Event 0 Disable. 1 A period event interrupt can be generated (EIR[TS_TIMER]) and the event signal output is asserted when the timer wraps around according to the periodic setting ATPER. The timer period value must be set before setting this bit. **NOTE:** Not all devices contain the event signal output. See the chip configuration details.

Table continues on the next page...

ENET_ATCR field descriptions (continued)

Field	Description
28 OFFRST	Reset Timer On Offset Event 0 The timer is not affected and no action occurs, besides clearing OFFEN, when the offset is reached. 1 If OFFEN is set, the timer resets to zero when the offset setting is reached. The offset event does not cause a timer interrupt.
29 OFFEN	Enable One-Shot Offset Event 0 Disable. 1 The timer can be reset to zero when the given offset time is reached (offset event). The field is cleared when the offset event is reached, so no further event occurs until the field is set again. The timer offset value must be set before setting this field.
30 Reserved	This field is reserved.
31 EN	Enable Timer 0 The timer stops at the current value. 1 The timer starts incrementing.

53.4.88 Timer Value Register (ENET_ATVR)

Address: 0h base + 404h offset = 404h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ATIME																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATVR field descriptions

Field	Description
0–31 ATIME	A write sets the timer. A read returns the last captured value. To read the current value, issue a capture command (i.e., set ATCR[CAPTURE]) prior to reading this register.

53.4.89 Timer Offset Register (ENET_ATOFF)

Address: 0h base + 408h offset = 408h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OFFSET																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATOFF field descriptions

Field	Description
0–31 OFFSET	Offset value for one-shot event generation. When the timer reaches the value, an event can be generated to reset the counter. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds.

53.4.90 Timer Period Register (ENET_ATPER)

Address: 0h base + 40Ch offset = 40Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PERIOD																															
Reset	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0

ENET_ATPER field descriptions

Field	Description
0–31 PERIOD	Value for generating periodic events. Each instance the timer reaches this value, the period event occurs and the timer restarts. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds. The value should be initialized to 1,000,000,000 (1×10^9) to represent a timer wrap around of one second. The increment value set in ATINC should be set to the true nanoseconds of the period of clock ts_clk, hence implementing a true 1 second counter. **NOTE:** The value of PERIOD has the following constraint: 2^{32} – ENET_ATINC[INC_COR] – 3×ENET_ATINC[INC] ≥ PERIOD > 0.

53.4.91 Timer Correction Register (ENET_ATCOR)

Address: 0h base + 410h offset = 410h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	COR														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	COR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATCOR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–31 COR	Correction Counter Wrap-Around Value

Table continues on the next page...

ENET_ATCOR field descriptions (continued)

Field	Description
	Defines after how many timer clock cycles (ts_clk) the correction counter should be reset and trigger a correction increment on the timer. The amount of correction is defined in ATINC[INC_CORR]. A value of 0 disables the correction counter and no corrections occur. **NOTE:** This value is given in clock cycles, not in nanoseconds as all other values.

53.4.92 Time-Stamping Clock Period Register (ENET_ATINC)

Address: 0h base + 414h offset = 414h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	INC_CORR							0	INC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATINC field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 INC_CORR	Correction Increment Value This value is added every time the correction timer expires (every clock cycle given in ATCOR). A value less than INC slows down the timer. A value greater than INC speeds up the timer.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 INC	Clock Period Of The Timestamping Clock (ts_clk) In Nanoseconds The timer increments by this amount each clock cycle. For example, set to 10 for 100 MHz, 8 for 125 MHz, 5 for 200 MHz. **NOTE:** For highest precision, use a value that is an integer fraction of the period set in ATPER.

53.4.93 Timestamp of Last Transmitted Frame (ENET_ATSTMP)

Address: 0h base + 418h offset = 418h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TIMESTAMP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATSTMP field descriptions

Field	Description
0–31 TIMESTAMP	Timestamp of the last frame transmitted by the core that had TxBD[TS] set . This register is only valid when EIR[TS_AVAIL] is set.

53.4.94 Timer Global Status Register (ENET_TGSR)

Address: 0h base + 604h offset = 604h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												TF3	TF2	TF1	TF0
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TGSR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TF3	Copy Of Timer Flag For Channel 3 0 Timer Flag for Channel 3 is clear 1 Timer Flag for Channel 3 is set
29 TF2	Copy Of Timer Flag For Channel 2 0 Timer Flag for Channel 2 is clear 1 Timer Flag for Channel 2 is set
30 TF1	Copy Of Timer Flag For Channel 1 0 Timer Flag for Channel 1 is clear 1 Timer Flag for Channel 1 is set
31 TF0	Copy Of Timer Flag For Channel 0 0 Timer Flag for Channel 0 is clear 1 Timer Flag for Channel 0 is set

53.4.95 Timer Control Status Register (ENET_TCSR*n*)

Address: 0h base + 608h offset + (8d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TF	TIE	TMODE				0	TDRE
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TCSR*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 TF	Timer Flag Sets when input capture or output compare occurs. This flag is double buffered between the module clock and 1588 clock domains. When this field is 1, it can be cleared to 0 by writing 1 to it. 0 Input Capture or Output Compare has not occurred. 1 Input Capture or Output Compare has occurred.
25 TIE	Timer Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
26–29 TMODE	Timer Mode Updating the Timer Mode field takes a few cycles to register because it is synchronized to the 1588 clock. The version of Timer Mode returned on a read is from the 1588 clock domain. When changing Timer Mode, always disable the channel and read this register to verify the channel is disabled first. 0000 Timer Channel is disabled. 0001 Timer Channel is configured for Input Capture on rising edge. 0010 Timer Channel is configured for Input Capture on falling edge. 0011 Timer Channel is configured for Input Capture on both edges. 0100 Timer Channel is configured for Output Compare - software only. 0101 Timer Channel is configured for Output Compare - toggle output on compare. 0110 Timer Channel is configured for Output Compare - clear output on compare. 0111 Timer Channel is configured for Output Compare - set output on compare. 1000 Reserved 1010 Timer Channel is configured for Output Compare - clear output on compare, set output on overflow. 10X1 Timer Channel is configured for Output Compare - set output on compare, clear output on overflow. 110X Reserved

Table continues on the next page...

ENET_TCSR*n* field descriptions (continued)

Field	Description
	1110 Timer Channel is configured for Output Compare - pulse output low on compare for one 1588-clock cycle. 1111 Timer Channel is configured for Output Compare - pulse output high on compare for one 1588-clock cycle.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDRE	Timer DMA Request Enable 0 DMA request is disabled 1 DMA request is enabled

53.4.96 Timer Compare Capture Register (ENET_TCCR*n*)

Address: 0h base + 60Ch offset + (8d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TCC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TCCR*n* field descriptions

Field	Description
0–31 TCC	Timer Capture Compare This register is double buffered between the module clock and 1588 clock domains. When configured for compare, the 1588 clock domain updates with the value in the module clock domain whenever the Timer Channel is first enabled and on each subsequent compare. Write to this register with the first compare value before enabling the Timer Channel. When the Timer Channel is enabled, write the second compare value either immediately, or at least before the first compare occurs. After each compare, write the next compare value before the previous compare occurs and before clearing the Timer Flag. The compare occurs one 1588 clock cycle after the IEEE 1588 Counter increments past the compare value in the 1588 clock domain. If the compare value is less than the value of the 1588 Counter when the Timer Channel is first enabled, then the compare does not occur until following the next overflow of the 1588 Counter. If the compare value is greater than the IEEE 1588 Counter when the 1588 Counter overflows, or the compare value is less than the value of the IEEE 1588 Counter after the overflow, then the compare occurs one 1588 clock cycle following the overflow. When configured for capture, the value of the IEEE 1588 Counter is captured into the 1588 clock domain and then updated into the module clock domain, provided the Timer Flag is clear. Always read the capture value before clearing the Timer Flag.

53.5 Functional description

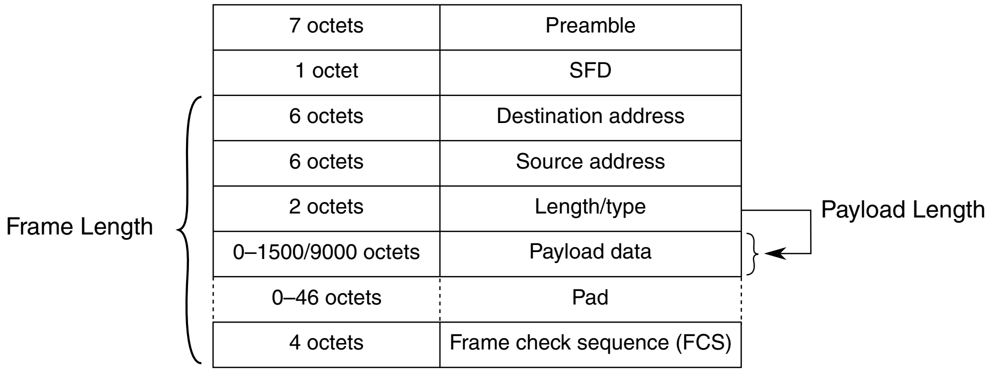
This section provides a complete functional description of the MAC-NET core.

53.5.1 Ethernet MAC frame formats

- Minimum length of 64 bytes
- Maximum length of 1518 bytes excluding the preamble and the start frame delimiter (SFD) bytes

An Ethernet frame consists of the following fields:

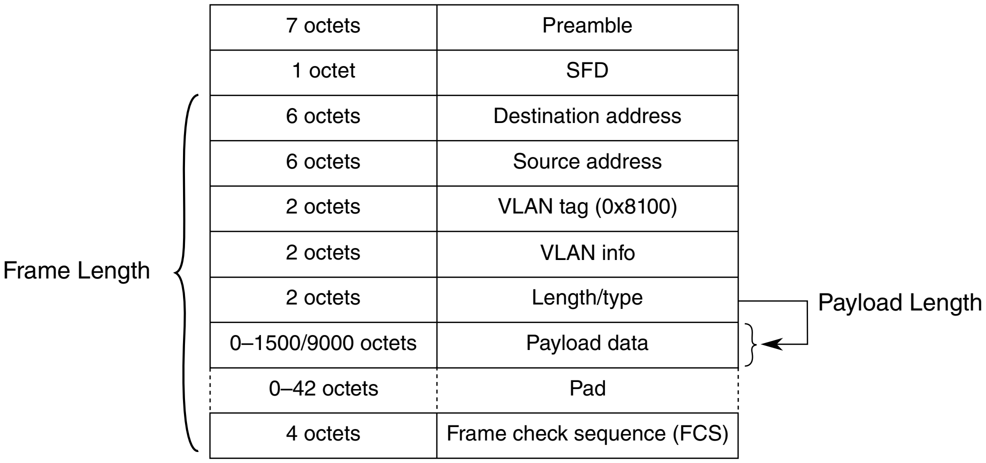
- Seven bytes preamble
- Start frame delimiter (SFD)
- Two address fields
- Length or type field
- Data field
- Frame check sequence (CRC value)



7 octets	Preamble
1 octet	SFD
6 octets	Destination address
6 octets	Source address
2 octets	Length/type
0–1500/9000 octets	Payload data
0–46 octets	Pad
4 octets	Frame check sequence (FCS)

Figure 53-2. MAC frame format overview

Optionally, MAC frames can be VLAN-tagged with an additional four-byte field inserted between the MAC source address and the type/length field. VLAN tagging is defined by the IEEE P802.1q specification. VLAN-tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes.



7 octets	Preamble
1 octet	SFD
6 octets	Destination address
6 octets	Source address
2 octets	VLAN tag (0x8100)
2 octets	VLAN info
2 octets	Length/type
0–1500/9000 octets	Payload data
0–42 octets	Pad
4 octets	Frame check sequence (FCS)

Figure 53-3. VLAN-tagged MAC frame format overview

Table 53-3. MAC frame definition

Term	Description
Frame length	Defines the length, in octets, of the complete frame without preamble and SFD. A frame has a valid length if it contains at least 64 octets and does not exceed the programmed maximum length.
Payload length	The length/type field indicates the length of the frame's payload section. The most significant byte is sent/received first. • If the length/type field is set to a value less than 46, the payload is padded so that the minimum frame length requirement (64 bytes) is met. For VLAN-tagged frames, a value less than 42 indicates a padded frame. • If the length/type field is set to a value larger than the programmed frame maximum length (e.g. 1518) it is interpreted as a type field.
Destination and source address	48-bit MAC addresses. The least significant byte is sent/received first and the first two least significant bits of the MAC address distinguish MAC frames, as detailed in MAC address check.

Note

Although the IEEE specification defines a maximum frame length, the MAC core provides the flexibility to program any value for the frame maximum length.

53.5.1.1 Pause Frames

The receiving device generates a pause frame to indicate a congestion to the emitting device, which should stop sending data.

Pause frames are indicated by the length/type set to 0x8808. The two first bytes of a pause frame following the type, defines a 16-bit opcode field set to 0x0001 always. A 16-bit pause quanta is defined in the frame payload bytes 2 (P1) and 3 (P2) as defined in the following table. The P1 pause quanta byte is the most significant.

Table 53-4. Pause Frame Format (Values in Hex)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
55	55	55	55	55	55	55	D5	01	80	C2	00	00	01
Preamble							SFD	Multicast Destination Address					
15	16	17	18	19	20	21	22	23	24	25	26	27 –68	
00	00	00	00	00	00	88	08	00	01	hi	lo	00	
Source Address						Type		Opcode		P1	P2	pad (42)	
69	70	71	72										
26	6B	AE	0A										
CRC-32													

There is no payload length field found within a pause frame and a pause frame is always padded with 42 bytes (0x00).

If a pause frame with a pause value greater than zero (XOFF condition) is received, the MAC stops transmitting data as soon the current frame transfer is completed. The MAC stops transmitting data for the value defined in pause quanta. One pause quanta fraction refers to 512 bit times.

If a pause frame with a pause value of zero (XON condition) is received, the transmitter is allowed to send data immediately (see Full-duplex flow control operation for details).

53.5.1.2 Magic packets

A magic packet is a unicast, multicast, or broadcast packet, which carries a defined sequence in the payload section.

Magic packets are received and inspected only under specific conditions.

The defined sequence to decode a magic packet is formed with a synchronization stream which consists of six consecutive 0xFF bytes, and is followed by sequence of sixteen consecutive unicast MAC addresses of the node to be awakened.

This sequence can be located anywhere in the magic packet payload. The magic packet is formed with a standard Ethernet header, optional padding, and CRC.

53.5.2 IP and higher layers frame format

For example, an IP datagram specifies the payload section of an Ethernet frame. A TCP datagram specifies the payload section within an IP datagram.

53.5.2.1 Ethernet types

IP datagrams are carried in the payload section of an Ethernet frame. The Ethernet frame type/length field discriminates several datagram types.

The following table lists the types of interest:

Table 53-5. Ethernet type value examples

Type	Description
0x8100	VLAN-tagged frame. The actual type is found 4 octets later in the frame.
0x0800	IPv4
0x0806	ARP
0x86DD	IPv6

53.5.2.2 IPv4 datagram format

The following figure shows the IP Version 4 (IPv4) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words. The first byte sent/received is the leftmost byte of the first word (in other words, version/IHL field).

The IP header can contain further options, which are always padded if necessary to guarantee the payload following the header is aligned to a 32-bit boundary.

The IP header is immediately followed by the payload, which can contain further protocol headers (for example, TCP or UDP, as indicated by the protocol field value). The complete IP datagram is transported in the payload section of an Ethernet frame.

Table 53-6. IPv4 header format

<table>
<tr><td colspan="4">31 30 29 28</td><td colspan="4">27 26 25 24</td><td colspan="4">23 22 21 20</td><td colspan="4">19 18 17 16</td><td colspan="4">15 14 13 12</td><td colspan="4">11 10 9 8</td><td colspan="4">7 6 5 4</td><td colspan="4">3 2 1 0</td></tr>
<tr><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="8">TOS</td><td colspan="16">Length</td></tr>
<tr><td colspan="16">Fragment ID</td><td colspan="3">Flags</td><td colspan="13">Fragment offset</td></tr>
<tr><td colspan="8">TTL</td><td colspan="8">Protocol</td><td colspan="16">Header checksum</td></tr>
<tr><td colspan="32">Source address</td></tr>
<tr><td colspan="32">Destination address</td></tr>
<tr><td colspan="32">Options</td></tr>
</table>

Table 53-7. IPv4 header fields

Field name	Description
Version	4-bit IP version information. 0x4 for IPv4 frames.
IHL	4-bit Internet header length information. Determines number of 32-bit words found within the IP header. If no options are present, the default value is 0x5.
TOS	Type of service/DiffServ field.
Length	Total length of the datagram in bytes, including all octets of header and payload.
Fragment ID, flags, fragment offset	Fields used for IP fragmentation.
TTL	Time-to-live. In effect, is decremented at each router arrival. If zero, datagram must be discarded.
Protocol	Identifier of protocol that follows in the datagram.
Header checksum	Checksum of IP header. For computational purposes, this field's value is zero.
Source address	Source IP address.
Destination address	Destination IP address.

53.5.2.3 IPv6 datagram format

The following figure shows the IP version 6 (IPv6) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words and has a fixed length of ten words (40 bytes). The next header field identifies the type of the header that follows the IPv6 header. It is defined similar to the protocol identifier within IPv4, with new definitions for identifying extension headers. These headers can be inserted between the IPv6 header and the protocol header, which will shift the protocol header accordingly.The accelerator currently only supports IPv6 without extension headers (in other words, the next header specifies TCP, UDP, or IMCP).

The first byte sent/received is the leftmost byte of the first word (in other words, version/ traffic class fields).

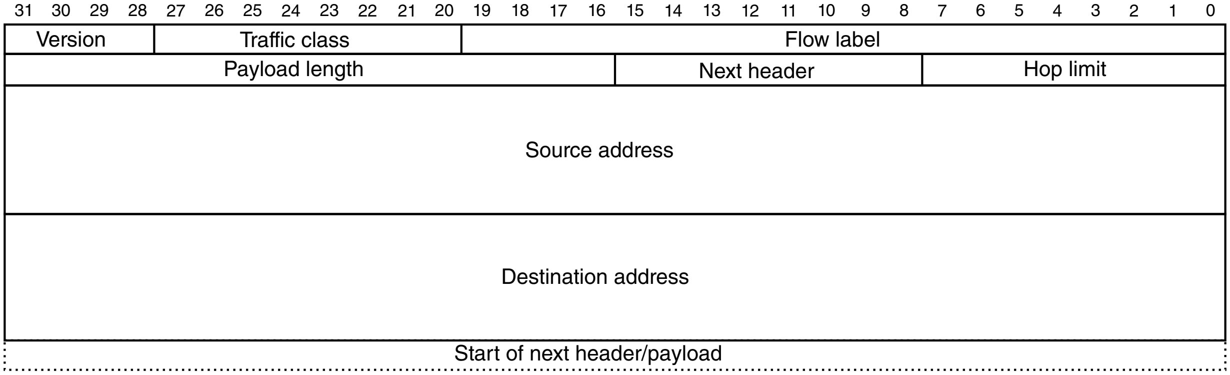


Figure 53-4. IPv6 header format

Table 53-8. IPv6 header fields

Field name	Description
Version	4-bit IP version information. 0x6 for all IPv6 frames.
Traffic class	8-bit field defining the traffic class.
Flow label	20-bit flow label identifying frames of the same flow.
Payload length	16-bit length of the datagram payload in bytes. It includes all octets following the IPv6 header.
Next header	Identifies the header that follows the IPv6 header. This can be the protocol header or any IPv6 defined extension header.
Hop limit	Hop counter, decremented by one by each station that forwards the frame. If hop limit is 0 the frame must be discarded.
Source address	128-bit IPv6 source address.
Destination address	128-bit IPv6 destination address.

53.5.2.4 Internet Control Message Protocol (ICMP) datagram format

An internet control message protocol (ICMP) is found following the IP header, if the protocol identifier is 1. The ICMP datagram has a four-octet header followed by additional message data.

Table 53-9. ICMP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Type		Code		Checksum			
ICMP message data							

Table 53-10. IP header fields

Field name	Description
Type	8-bit type information
Code	8-bit code that is related to the message type
Checksum	16-bit one's complement checksum over the complete ICMP datagram

53.5.2.5 User Datagram Protocol (UDP) datagram format

A user datagram protocol header is found after the IP header, when the protocol identifier is 17.

The payload of the datagram is after the UDP header. The header byte order follows the conventions given for the IP header above.

Table 53-11. UDP header format

<table>
<tr><td colspan="4">31 30 29 28</td><td colspan="4">27 26 25 24</td><td colspan="4">23 22 21 20</td><td colspan="4">19 18 17 16</td><td colspan="4">15 14 13 12</td><td colspan="4">11 10 9 8</td><td colspan="4">7 6 5 4</td><td colspan="4">3 2 1 0</td></tr>
<tr><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><td colspan="16">Length</td><td colspan="16">Checksum</td></tr>
</table>

Table 53-12. UDP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Length	Length of user data which immediately follows the header, including the UDP header (that is, minimum value is 8)
Checksum	Checksum over the complete datagram and some IP header information

53.5.2.6 TCP datagram format

A TCP header is found following the IP header, when the protocol identifier has a value of 6.

The TCP payload immediately follows the TCP header.

Table 53-13. TCP header format

<table>
<tr><td colspan="4">31 30 29 28</td><td colspan="4">27 26 25 24</td><td colspan="4">23 22 21 20</td><td colspan="4">19 18 17 16</td><td colspan="4">15 14 13 12</td><td colspan="4">11 10 9 8</td><td colspan="4">7 6 5 4</td><td colspan="4">3 2 1 0</td></tr>
<tr><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><td colspan="32">Sequence number</td></tr>
<tr><td colspan="32">Acknowledgement number</td></tr>
<tr><td colspan="4">Offset</td><td colspan="6">Reserved</td><td colspan="6">Flags</td><td colspan="16">Window</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Urgent pointer</td></tr>
<tr><td colspan="32">Options</td></tr>
</table>

Table 53-14. TCP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Sequence number	Transmit sequence number
Ack. number	Receive sequence number
Offset	Data offset, which is number of 32-bit words within TCP header — if no options selected, defaults to value of 5

Table continues on the next page...

Table 53-14. TCP header fields (continued)

Field name	Description
Flags	URG, ACK, PSH, RST, SYN, FIN flags
Window	TCP receive window size information
Checksum	Checksum over the complete datagram (TCP header and data) and IP header information
Options	Additional 32-bit words for protocol options

53.5.3 IEEE 1588 message formats

The following sections describe the IEEE 1588 message formats.

53.5.3.1 Transport encapsulation

The precision time protocol (PTP) datagrams are encapsulated in Ethernet frames using the UDP/IP transport mechanism, or optionally, with the newer 1588v2 directly in Ethernet frames (layer 2).

Typically, multicast addresses are used to allow efficient distribution of the synchronization messages.

53.5.3.1.1 UDP/IP

The 1588 messages (v1 and v2) can be transported using UDP/IP multicast messages.

Table 53-15 shows IP multicast groups defined for PTP. The table also shows their respective MAC layer multicast address mapping according to RFC 1112 (last three octets of IP follow the fixed value of 01-00-5E).

Table 53-15. UDP/IP multicast domains

Name	IP Address	MAC Address mapping
DefaultPTPdomain	224.0.1.129	01-00-5E-00-01-81
AlternatePTPdomain1	224.0.1.130	01-00-5E-00-01-82
AlternatePTPdomain2	224.0.1.131	01-00-5E-00-01-83
AlternatePTPdomain3	224.0.1.132	01-00-5E-00-01-84

Table 53-16. UDP port numbers

Message type	UDP port	Note
Event	319	Used for SYNC and DELAY_REQUEST messages
General	320	All other messages (for example, follow-up, delay-response)

53.5.3.1.2 Native Ethernet (PTPv2)

In addition to using UDP/IP frames, IEEE 1588v2 defines a native Ethernet frame format that uses ethertype = 0x88F7. The payload of the Ethernet frame immediately contains the PTP datagram, starting with the PTPv2 header.

Besides others, version 2 adds a peer delay mechanism to allow delay measurements between individual point-to-point links along a path over multiple nodes. The following multicast domains are also defined in PTPv2.

Table 53-17. PTPv2 multicast domains

Name	MAC address
Normal messages	01-1B-19-00-00-00
Peer delay messages	01-80-C2-00-00-0E

53.5.3.2 PTP header

All PTP frames contain a common header that determines the protocol version and the type of message, which defines the remaining content of the message.

All multi-octet fields are transmitted in big-endian order (the most significant byte is transmitted/received first).

The last four bits of versionPTP are at the same position (second byte) for PTPv1 and PTPv2 headers. This allows accurate identification by inspecting the first two bytes of the message.

53.5.3.2.1 PTPv1 header

Table 53-18. Common PTPv1 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	2	versionPTP = 0x0001							
2	2	versionNetwork							
4	16	subdomain							
20	1	messageType							
21	1	sourceCommunicationTechnology							
22	6	sourceUuid							
28	2	sourcePortId							
30	2	sequenceId							

Table continues on the next page...

Table 53-18. Common PTPv1 message header (continued)

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
32	1	control							
33	1	0x00							
34	2	flags							
36	4	reserved							

The type of message is encoded in the messageType and control fields as shown in Table 53-19 :

Table 53-19. PTPv1 message type identification

messageType	control	Message Name	Message
0x01	0x0	SYNC	Event message
0x01	0x1	DELAY_REQ	Event message
0x02	0x2	FOLLOW_UP	General message
0x02	0x3	DELAY_RESP	General message
0x02	0x4	MANAGEMENT	General message
other	other	—	Reserved

The field sequenceId is used to non-ambiguously identify a message.

53.5.3.2.2 PTPv2 header

Table 53-20. Common PTPv2 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	1	transportSpecific				messageId			
1	1	reserved				versionPTP = 0x2			
2	2	messageLength							
4	1	domainNumber							
5	1	reserved							
6	2	flags							
8	8	correctionField							
16	4	reserved							
20	10	sourcePortIdentity							
30	2	sequenceId							
32	1	control							
33	1	logMeanMessageInterval							

The type of message is encoded in the field messageId as follows:

Table 53-21. PTPv2 message type identification

messageId	Message name	Message
0x0	SYNC	Event message
0x1	DELAY_REQ	Event message
0x2	PATH_DELAY_REQ	Event message
0x3	PATH_DELAY_RESP	Event message
0x4–0x7	—	Reserved
0x8	FOLLOW_UP	General message
0x9	DELAY_RESP	General message
0xa	PATH_DELAY_FOLLOW_UP	General message
0xb	ANNOUNCE	General message
0xc	SIGNALING	General message
0xd	MANAGEMENT	General message

The PTPv2 flags field contains further details on the type of message, especially if one-step or two-step implementations are used. The one- or two-step implementation is controlled by the TWO_STEP bit in the first octet of the flags field as shown below. Reserved bits are cleared.

Table 53-22. PTPv2 message flags field definitions

Bit	Name	Description
0	ALTERNATE_MASTER	See IEEE 1588 Clause 17.4
1	TWO_STEP	1 Two-step clock 0 One-step clock
2	UNICAST	1 Transport layer address uses a unicast destination address 0 Multicast is used
3	—	Reserved
4	—	Reserved
5	Profile specific	
6	Profile specific	
7	—	Reserved

53.5.4 MAC receive

- Check frame framing
- Remove frame preamble and frame SFD field

- Discard frame based on frame destination address field
- Terminate pause frames
- Check frame length
- Remove payload padding if it exists
- Calculate and verify CRC-32
- Write received frames in the core receive FIFO

If the MAC is programmed to operate in half-duplex mode, it will also check if the frame is received with a collision.

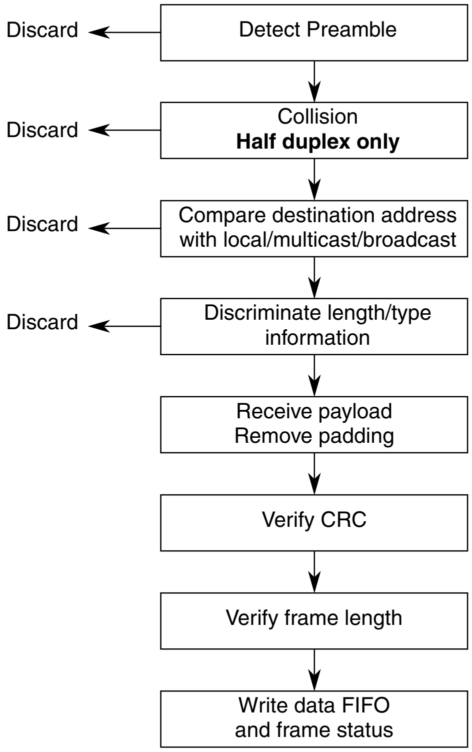


Figure 53-5. MAC receive flow

53.5.4.1 Collision detection in half-duplex mode

If the packet is received with a collision detected during reception of the first 64 bytes, the packet is discarded (if frame size was less than ~14 octets) or transmitted to the user application with an error and RxBD[CE] set.

53.5.4.2 Preamble processing

The IEEE 802.3 standard allows a maximum size of 56 bits (seven bytes) for the preamble, while the MAC core allows any preamble length, including zero length preamble.

The MAC core checks for the start frame delimiter (SFD) byte. If the next byte of the preamble, which is different from 0x55, is not 0xD5, the frame is discarded.

Although the IEEE specification dictates that the inner-packet gap should be at least 96 bits, the MAC core is designed to accept frames separated by only 64 10/100-Mbit/s operation (MII) bits.

The MAC core removes the preamble and SFD bytes.

53.5.4.3 MAC address check

The destination address bit 0 differentiates between multicast and unicast addresses.

- If bit 0 is 0, the MAC address is an individual (unicast) address.
- If bit 0 is 1, the MAC address defines a group (multicast) address.
- If all 48 bits of the MAC address are set, it indicates a broadcast address.

53.5.4.3.1 Unicast address check

If a unicast address is received, the destination MAC address is compared to the node MAC address programmed by the host in the PADDR1/2 registers.

If the destination address matches any of the programmed MAC addresses, the frame is accepted.

If Promiscuous mode is enabled (RCR[PROM] = 1) no address checking is performed and all unicast frames are accepted.

53.5.4.3.2 Multicast and unicast address resolution

The hash table algorithm used in the group and individual hash filtering operates as follows.

- The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits in ENET*n*_GAUR/GALR (group address hash match) or ENET*n*_IAUR/IALR (individual address hash match).

- This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63.
- The msb of the CRC result selects ENET*n*_GAUR (msb = 1) or ENET*n*_GALR (msb = 0).
- The five lsbs of the hash result select the bit within the selected register.
- If the CRC generator selects a bit set in the hash table, the frame is accepted; else, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

- $FCS(x) = x^{32}+ x^{26}+ x^{23}+ x^{22}+ x^{16}+ x^{12}+ x^{11}+ x^{10}+ x^{8}+ x^{7}+ x^{5}+ x^{4}+ x^{2}+ x^{1}+ 1$

If Promiscuous mode is enabled (ENET*n*_RCR[PROM] = 1) all unicast and multicast frames are accepted regardless of ENET*n*_GAUR/GALR and ENET*n*_IAUR/IALR settings.

53.5.4.3.3 Broadcast address reject

All broadcast frames are accepted if BC_REJ is cleared or ENET*n*_RCR[PROM] is set. If PROM is cleared when ENET*n*_RCR[BC_REJ] is set, all broadcast frames are rejected.

Table 53-23. Broadcast address reject programming

PROM	BC_REJ	Broadcast frames
0	0	Accepted
0	1	Rejected
1	0	Accepted
1	1	Accepted

53.5.4.3.4 Miss-bit implementation

For higher layer filtering purposes, RxBD[M] indicates an address miss when the MAC operates in promiscuous mode and accepts a frame that would otherwise be rejected.

If a group/individual hash or exact match does not occur and Promiscuous mode is enabled (RCR[PROM] = 1), the frame is accepted and the M bit is set in the buffer descriptor; otherwise, the frame is rejected.

This means the status bit is set in any of the following conditions during Promiscuous mode:

- A broadcast frame is received when BC_REJ is set
- A unicast is received that does not match either:
  - Node address (PALR[PADDR1] and PAUR[PADDR2])
  - Hash table for unicast (IAUR[IADDR1] and IALR[IADDR2])
- A multicast is received that does not match the GAUR[GADDR1] and GALR[GADDR2] hash table entries

53.5.4.4 Frame length/type verification: payload length check

If the length/type is less than 0x600 and NLC is set, the MAC checks the payload length and reports any error in the frame status word and interrupt bit PLR.

If the length/type is greater than or equal to 0x600, the MAC interprets the field as a type and no payload length check is performed.

The length check is performed on VLAN and stacked VLAN frames. If a padded frame is received, no length check can be performed due to the extended frame payload because padded frames can never have a payload length error.

53.5.4.5 Frame length/type verification: frame length check

When the receive frame length exceeds MAX_FL bytes, the BABR interrupt is generated and the RxBD[LG] bit is set.

The frame is not truncated unless the frame length exceeds the value programmed in ENET*n*_FTRL[TRUNC_FL]. If the frame is truncated, RxBD[TR] is set. In addition, a truncated frame always has the CRC error indication set (RxBD[CR]).

53.5.4.6 VLAN frames processing

VLAN frames have a length/type field set to 0x8100 immediately followed by a 16-Bit VLAN control information field.

VLAN-tagged frames are received as normal frames because the VLAN tag is not interpreted by the MAC function, and are pushed complete with the VLAN tag to the user application. If the length/type field of the VLAN-tagged frame, which is found four octets later in the frame, is less than 42, the padding is removed. In addition, the frame status word (RxBD[NO]) indicates that the current frame is VLAN tagged.

53.5.4.7 Pause frame termination

The receive engine terminates pause frames and does not transfer them to the receive FIFO. The quanta is extracted and sent to the MAC transmit path via a small internal clock rate decoupling asynchronous FIFO.

The quanta is written only if a correct CRC and frame length are detected by the control state machine. If not, the quanta is discarded and the MAC transmit path is not paused.

Good pause frames are ignored if ENET*n*_RCR[FCE] is cleared and are forwarded to the client interface when ENET*n*_RCR[PAUFWD] is set.

53.5.4.8 CRC check

The CRC-32 field is checked and forwarded to the core FIFO interface if ENET*n*_RCR[CRCFWD] is cleared and ENET*n*_RCR[PADEN] is set.

When CRCFWD is set (regardless of PADEN), the CRC-32 field is checked and terminated (not transmitted to the FIFO).

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32}+ x^{26}+ x^{23}+ x^{22}+ x^{16}+ x^{12}+ x^{11}+ x^{10}+ x^{8}+ x^{7}+ x^{5}+ x^{4}+ x^{2}+ x^{1}+ 1$

The 32 bits of the CRC value are placed in the frame check sequence (FCS) field with the x^{31} term as right-most bit of the first octet. The CRC bits are thus received in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

If a CRC error is detected, the frame is marked invalid and RxBD[CR] is set.

53.5.4.9 Frame padding removal

When a frame is received with a payload length field set to less than 46 (42 for VLAN-tagged frames and 38 for frames with stacked VLANs), the zero padding can be removed before the frame is written into the data FIFO depending on the setting of ENET*n*_RCR[PADEN].

Note

> If a frame is received with excess padding (in other words, the length field is set as mentioned above, but the frame has more than 64 octets) and padding removal is enabled, then the padding is removed as normal and no error is reported if the frame is otherwise correct (for example: good CRC, less than maximum length, and no other error).

53.5.5 MAC transmit

Frame transmission starts when the transmit FIFO holds enough data.

After a transfer starts, the MAC transmit function performs the following tasks:

- Generates preamble and SFD field before frame transmission
- Generates XOFF pause frames if the receive FIFO reports a congestion or if ENET*n*_TCR[TFC_PAUSE] is set with ENET*n*_OPD[PAUSE_DUR] set to a non-zero value
- Generates XON pause frames if the receive FIFO congestion condition is cleared or if TFC_PAUSE is set with PAUSE_DUR cleared
- Suspends Ethernet frame transfer (XOFF) if a non-zero pause quanta is received from the MAC receive path
- Adds padding to the frame if required
- Calculates and appends CRC-32 to the transmitted frame
- Sends the frame with correct inter-packet gap (IPG) (deferring)

When the MAC is configured to operate in half-duplex mode, the following additional tasks are performed:

- Collision detection
- Frame retransmit after back-off timer expires

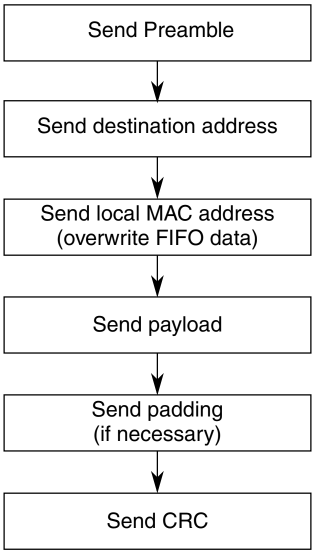

Send Preamble
Send destination address
Send local MAC address (overwrite FIFO data)
Send payload
Send padding (if necessary)
Send CRC

Figure 53-6. Frame transmit overview

53.5.5.1 Frame payload padding

The IEEE specification defines a minimum frame length of 64 bytes.

If the frame sent to the MAC from the user application has a size smaller than 60 bytes, the MAC automatically adds padding bytes (0x00) to comply with the Ethernet minimum frame length specification. Transmit padding is always performed and cannot be disabled.

If the MAC is not allowed to append a CRC (TxBD[TC] = 1), the user application is responsible for providing frames with a minimum length of 64 octets.

53.5.5.2 MAC address insertion

On each frame received from the core transmit FIFO interface, the source MAC address is either:

- Replaced by the address programmed in the PADDR1/2 fields (ENET*n*_TCR[ADDINS] = 1)
- Transparently forwarded to the Ethernet line (ENET*n*_TCR[ADDINS] = 0)

53.5.5.3 CRC-32 generation

The CRC-32 field is optionally generated and appended at the end of a frame.

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32}+ x^{26}+ x^{23}+ x^{22}+ x^{16}+ x^{12}+ x^{11}+ x^{10}+ x^{8}+ x^{7}+ x^{5}+ x^{4}+ x^{2}+ x^{1}+ 1$

The 32 bits of the CRC value are placed in the FCS field so that the x^{31} term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

53.5.5.4 Inter-packet gap (IPG)

In full-duplex mode, after frame transmission and before transmission of a new frame, an IPG (programmed in ENET*n*_TIPG) is maintained. The minimum IPG can be programmed between 8 and 26 byte-times (64 and 208 bit-times).

In half-duplex mode, the core constantly monitors the line. Actual transmission of the data onto the network occurs only if it has been idle for a 96-bit time period, and any back-off time requirements have been satisfied. In accordance with the standard, the core begins to measure the IPG from CRS de-assertion.

53.5.5.5 Collision detection and handling — half-duplex operation only

A collision occurs on a half-duplex network when concurrent transmissions from two or more nodes take place. During transmission, the core monitors the line condition and detects a collision when the PHY device asserts COL.

When the core detects a collision while transmitting, it stops transmission of the data and transmits a 32-bit jam pattern. If the collision is detected during the preamble or the SFD transmission, the jam pattern is transmitted after completing the SFD, which results in a minimum 96-bit fragment. The jam pattern is a fixed pattern that is not compared to the actual frame CRC, and has a very low probability (0.532) of having a jam pattern identical to the CRC.

If a collision occurs before transmission of 64 bytes (including preamble and SFD), the MAC core waits for the backoff period and retransmits the packet data (stored in a 64-byte re-transmit buffer) that has already been sent on the line. The backoff period is generated from a pseudo-random process (truncated binary exponential backoff).

If a collision occurs after transmission of 64 bytes (including preamble and SFD), the MAC discards the remainder of the frame, optionally sets the LC interrupt bit, and sets TxBD[LCE].

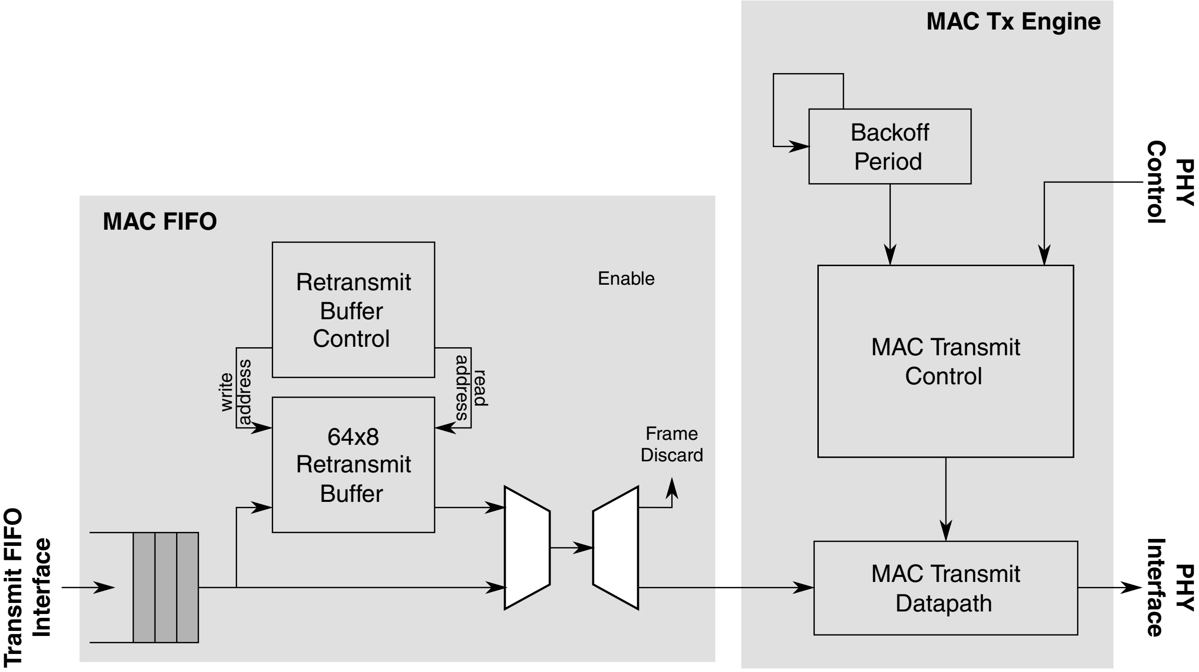


Figure 53-7. Packet re-transmit overview

The backoff time is represented by an integer multiple of slot times. One slot is equal to a 512-bit time period. The number of the delay slot times, before the *n*th re-transmission attempt, is chosen as a uniformly-distributed random integer in the range:

- $0 < r < 2^k$
- k = min(*n*, N); where *n* is the number of retransmissions and N = 10

For example, after the first collision, the backoff period is 0 or 1 slot time. If a collision occurs on the first retransmission, the backoff period is 0, 1, 2, or 3, and so on.

The maximum backoff time (in 512-bit time slots) is limited by N = 10 as specified in the IEEE 802.3 standard.

If a collision occurs after 16 consecutive retransmissions, the core reports an excessive collision condition (ENET*n*_EIR[RL] interrupt field and TxBD[EE]) and discards the current packet from the FIFO.

In networks violating the standard requirements, a collision may occur after transmission of the first 64 bytes. In this case, the core stops the current packet transmission and discards the rest of the packet from the transmit FIFO. The core resumes transmission with the next packet available in the core transmit FIFO.

warning

> Ethernet PHYs that support the SQE Test, or "heartbeat," feature must disable this feature. When this feature is enabled, the PHY asserts the collision signal after a frame is transmitted to indicate to the ENET that the PHY's collision logic is working. This may cause data corruption in the next frame from the ENET. This corrupted frame contains up to 21 zero bytes which start somewhere within the MAC destination address field. The ENET, however, will still generate a good FCS (CRC-32) but with corrupted data.

53.5.6 Full-duplex flow control operation

Three conditions are handled by the core's flow control engine:

- Remote device congestion — The remote device connected to the same Ethernet segment as the core reports congestion and requests that the core stop sending data.
- Core FIFO congestion — When the core's receive FIFO reaches a user-programmable threshold (RX section empty), the core sends a pause frame back to the remote device requesting the data transfer to stop.
- Local device congestion — Any device connected to the core can request (typically, via the host processor) the remote device to stop transmitting data.

53.5.6.1 Remote device congestion

When the MAC transmit control gets a valid pause quanta from the receive path and if ENET*n*_RCR[FCE] is set, the MAC transmit logic:

- Completes the transfer of the current frame.
- Stops sending data for the amount of time specified by the pause quanta in 512 bit time increments.
- Sets ENET*n*_TCR[RFC_PAUSE].

Frame transfer resumes when the time specified by the quanta expires and if no new quanta value is received, or if a new pause frame with a quanta value set to 0x0000 is received. The MAC also resets RFC_PAUSE to zero.

If ENET*n*_RCR[FCE] cleared, the MAC ignores received pause frames.

Optionally and independent of ENET*n*_RCR[FCE], pause frames are forwarded to the client interface if PAUFWD is set.

53.5.6.2 Local device/FIFO congestion

The MAC transmit engine generates pause frames when the local receive FIFO is not able to receive more than a pre-defined number of words (FIFO programmable threshold) or when pause frame generation is requested by the local host processor.

- To generate a pause frame, the host processor sets ENET*n*_TCR[TFC_PAUSE]. A single pause frame is generated when the current frame transfer is completed and TFC_PAUSE is automatically cleared. Optionally, an interrupt is generated.
- An XOFF pause frame is generated when the receive FIFO asserts its section empty flag (internal). An XOFF pause frame is generated automatically, when the current frame transfer completes.
- An XON pause frame is generated when the receive FIFO deasserts its section empty flag (internal). An XON pause frame is generated automatically, when the current frame transfer completes.

When an XOFF pause frame is generated, the pause quanta (payload byte P1 and P2) is filled with the value programmed in ENET*n*_OPD[PAUSE_DUR].

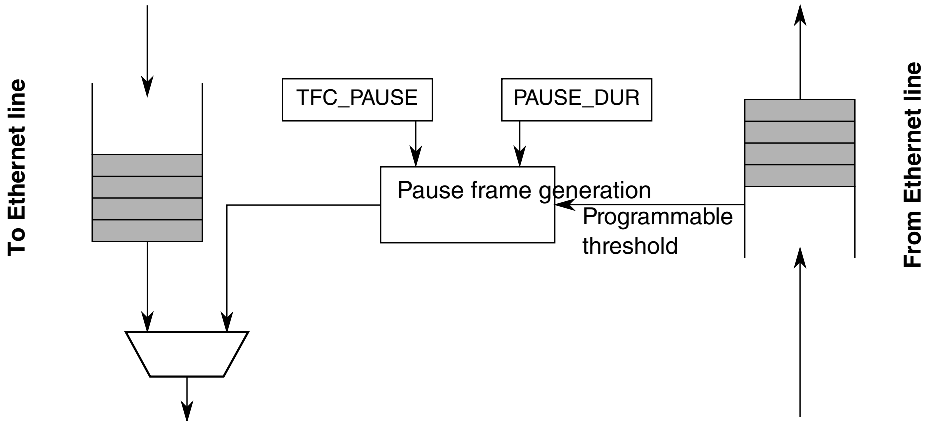


Figure 53-8. Pause frame generation overview

Note

Although the flow control mechanism should prevent any FIFO overflow on the MAC core receive path, the core receive FIFO is protected. When an overflow is detected on the receive FIFO, the current frame is truncated with an error indication set in the frame status word. The frame should subsequently be discarded by the user application.

53.5.7 Magic packet detection

Magic packet detection wakes a node that is put in power-down mode by the node management agent. Magic packet detection is supported only if the MAC is configured in sleep mode.

53.5.7.1 Sleep mode

To put the MAC in Sleep mode, set ENET*n*_ECR[SLEEP]. At the same time ENET*n*_ECR[MAGICEN] should also be set to enable magic packet detection.

In addition, when the processor is in Stop mode, Sleep mode is entered, without affecting the ENET*n*_ECR register bits.

When the MAC is in Sleep mode:

- The transmit logic is disabled.
- The FIFO receive/transmit functions are disabled.
- The receive logic is kept in Normal mode, but it ignores all traffic from the line except magic packets. They are detected so that a remote agent can wake the node.

53.5.7.2 Magic packet detection

The core is designed to detect magic packets with the destination address set to:

- Any multicast address
- The broadcast address
- The unicast address programmed in PADDR1/2

When a magic packet is detected, EIR[WAKEUP] is set and none of the statistic registers are incremented.

53.5.7.3 Wakeup

When a magic packet is detected, indicated by ENET*n*_EIR[WAKEUP], ENET*n*_ECR[SLEEP] should be cleared to resume normal operation of the MAC. Clearing the SLEEP bit automatically masks ENET*n*_ECR[MAGICEN], disabling magic packet detection.

53.5.8 IP accelerator functions

53.5.8.1 Checksum calculation

The IP and ICMP, TCP, UDP checksums are calculated with one's complement arithmetic summing up 16-bit values.

- For ICMP, the checksum is calculated over the complete ICMP datagram, in other words without IP header.
- For TCP and UDP, the checksums contain the header and data sections and values from the IP header, which can be seen as a pseudo-header that is not actually present in the data stream.

Table 53-24. IPv4 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
Zero		Protocol		TCP/UDP length			

Table 53-25. IPv6 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
TCP/UDP length							
Zero						Next header	

The TCP/UDP length value is the length of the TCP or UDP datagram, which is equal to the payload of an IP datagram. It is derived by subtracting the IP header length from the complete IP datagram length that is given in the IP header (IPv4), or directly taken from the IP header (IPv6). The protocol field is the corresponding value from the IP header. The Zero fields are all zeroes.

For IPv6, the complete 128-bit addresses are considered. The next header value identifies the upper layer protocol as either TCP or UDP. It may differ from the next header value of the IPv6 header if extension headers are inserted before the protocol header.

The checksum calculation uses 16-bit words in network byte order: The first byte sent/ received is the MSB, and the second byte sent/received is the LSB of the 16-bit value to add to the checksum. If the frame ends on an odd number of bytes, a zero byte is appended for checksum calculation only, and is not actually transmitted.

53.5.8.2 Additional padding processing

According to IEEE 802.3, any Ethernet frame must have a minimum length of 64 octets.

The MAC usually removes padding on receive when a frame with length information is received. Because IP frames have a type value instead of length, the MAC does not remove padding for short IP frames, as it is not aware of the frame contents.

The IP accelerator function can be configured to remove the Ethernet padding bytes that might follow the IP datagram.

On transmit, the MAC automatically adds padding as necessary to fill any frame to a 64-byte length.

53.5.8.3 32-bit Ethernet payload alignment

The data FIFOs allow inserting two additional arbitrary bytes in front of a frame. This extends the 14-byte Ethernet header to a 16-byte header, which leads to alignment of the Ethernet payload, following the Ethernet header, on a 32-bit boundary.

This function can be enabled for transmit and receive independently with the corresponding SHIFT16 bits in the ENET*n*_TACC and ENET*n*_RACC registers.

When enabled, the valid frame data is arranged as shown in Table 53-26.

Table 53-26. 64-bit interface data structure with SHIFT16 enabled

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Any value	Any value
Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6
...							

53.5.8.3.1 Receive processing

When ENET*n*_RACC[SHIFT16] is set, each frame is received with two additional bytes in front of the frame.

The user application must ignore these first two bytes and find the first byte of the frame in bits 23–16 of the first word from the RX FIFO.

Note

> SHIFT16 must be set during initialization and kept set during the complete operation, because it influences the FIFO write behavior.

53.5.8.3.2 Transmit processing

When ENET*n*_TACC[SHIFT16] is set, the first two bytes of the first word written (bits 15–0) are discarded immediately by the FIFO write logic.

The SHIFT16 bit can be enabled/disabled for each frame individually if required, but can be changed only between frames.

53.5.8.4 Received frame discard

Because the receive FIFO must be operated in store and forward mode (ENET*n*_RSFL cleared), received frames can be discarded based on the following errors:

- The MAC function receives the frame with an error:
  - The frame has an invalid payload length
  - Frame length is greater than MAX_FL
  - Frame received with a CRC-32 error
  - Frame truncated due to receive FIFO overflow
  - Frame is corrupted as PHY signaled an error (RX_ERR asserted during reception)
- An IP frame is detected and the IP header checksum is wrong
- An IP frame with a valid IP header and a valid IP header checksum is detected, the protocol is known but the protocol-specific checksum is wrong

If one of the errors occurs and the IP accelerator function is configured to discard frames (ENET*n*_RACC), the frame is automatically discarded. Statistics are maintained normally and are not affected by this discard function.

53.5.8.5 IPv4 fragments

When an IPv4 IP fragment frame is received, only the IP header is inspected and its checksum verified. 32-bit alignment operates the same way on fragments as it does on normal IP frames.

The IP fragment frame payload is not inspected for any protocol headers. As such, a protocol header would only exist in the very first fragment. To assist in protocol-specific checksum verification, the one's-complement sum is calculated on the IP payload (all bytes following the IP header) and provided with the frame status word.

The frame fragment status field (RxBD[FRAG]) is set to indicate a fragment reception, and the one's-complement sum of the IP payload is available in RxBD[Payload checksum].

Note

> After all fragments have been received and reassembled, the application software can take advantage of the payload checksum delivered with the frame's status word to calculate the protocol-specific checksum of the datagram.
>
> For example, if a TCP payload is delivered by multiple IP fragments, the application software can calculate the pseudo-header checksum value from the first fragment, and add the payload checksums delivered with the status for all fragments to verify the TCP datagram checksum.

53.5.8.6 IPv6 support

The following sections describe the IPv6 support.

53.5.8.6.1 Receive processing

An Ethernet frame of type 0x86DD identifies an IP Version 6 frame (IPv6) frame. If an IPv6 frame is received, the first IP header is inspected (first ten words), which is available in every IPv6 frame.

If the receive SHIFT16 function is enabled, the IP header is aligned on a 32-bit boundary allowing more efficient processing.

For TCP and UDP datagrams, the pseudo-header checksum calculation is performed and verified.

To assist in protocol-specific checksum verification, the one's-complement sum is always calculated on the IP payload (all bytes following the IP header) and provided with the frame status word. For example, if extension headers were present, their sums can be subtracted in software from the checksum to isolate the TCP/UDP datagram checksum, if required.

53.5.8.6.2 Transmit processing

For IPv6 transmission, the SHIFT16 function is supported to process 32-bit aligned datagrams.

IPv6 has no IP header checksum; therefore, the IP checksum insertion configuration is ignored.

The protocol checksum is inserted only if the next header of the IP header is a known protocol (TCP, UDP, or ICMP). If a known protocol is detected, the checksum over all bytes following the IP header is calculated and inserted in the correct position.

The pseudo-header checksum calculation is performed for TCP and UDP datagrams accordingly.

53.5.9 Resets and stop controls

53.5.9.1 Hardware reset

To reset the Ethernet module, set ENET*n*_ECR[RESET].

53.5.9.2 Soft reset

When ENET*n*_ECR[ETHER_EN] is cleared during operation, the following occurs:

- uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers.
- A currently ongoing transmit is terminated by asserting TXER to the PHY.

- A currently ongoing transmit FIFO write from the application is terminated by stopping the write to the FIFO, and all further data from the application is ignored. All subsequent writes are ignored until re-enabled.
- A currently ongoing receive FIFO read is terminated. The RxBD has arbitrary values in this case.

53.5.9.3 Hardware freeze

When the processor enters debug mode and ECR[DBGEN] is set, the MAC enters a freeze state where it stops all transmit and receive activities gracefully.

The following happens when the MAC enters hardware freeze:

- A currently ongoing receive transaction on the receive application interface is completed as normal. No further frames are read from the FIFO.
- A currently ongoing transmit transaction on the transmit application interface is completed as normal (in other words, until writing end-of-packet (EOP)).
- A currently ongoing frame receive is completed normally, after which no further frames are accepted from the MII.
- A currently ongoing frame transmit is completed normally, after which no further frames are transmitted.

53.5.9.4 Graceful stop

During a graceful stop, any currently ongoing transactions are completed normally and no further frames are accepted. The MAC can resume from a graceful stop without the need for a reset (for example, clearing ETHER_EN is not required).

The following conditions lead to a graceful stop of the MAC transmit or receive datapaths.

53.5.9.4.1 Graceful transmit stop (GTS)

When gracefully stopped, the MAC is no longer reading frame data from the transmit FIFO and has completed any ongoing transmission.

In any of the following conditions, the transmit datapath stops after an ongoing frame transmission has been completed normally.

- ENET*n*_TCR[GTS] is set by software.
- ENET*n*_TCR[TFC_PAUSE] is set by software requesting a pause frame transmission. The status (and register bit) is cleared after the pause frame has been sent.
- A pause frame was received stopping the transmitter. The stopped situation is terminated when the pause timer expires or a pause frame with zero quanta is received.
- MAC is placed in Sleep mode by software or the processor entering Stop mode (see Sleep mode).
- The MAC is in Hardware Freeze mode.

When the transmitter has reached its stopped state, the following events occur:

- The GRA interrupt is asserted, when transitioned into stopped.
- In Hardware Freeze mode, the GRA interrupt does not wait for the application write completion and asserts when the transmit state machine (in other words, line side of TX FIFO) reaches its stopped state.

53.5.9.4.2 Graceful receive stop (GRS)

When gracefully stopped, the MAC is no longer writing frames into the receive FIFO.

The receive datapath stops after any ongoing frame reception has been completed normally, if any of the following conditions occur:

- MAC is placed in Sleep mode either by the software or the processor is in Stop mode). The MAC continues to receive frames and search for magic packets if enabled (see Magic packet detection). However, no frames are written into the receive FIFO, and therefore are not forwarded to the application.
- The MAC is in Hardware Freeze mode. The MAC does not accept any frames from the MII.

When the receive datapath is stopped, the following events occur:

- If the RX is in the stopped state, RCR[GRS] is set

- The GRA interrupt is asserted when the transmitter and receiver are stopped
- Any ongoing receive transaction to the application (RX FIFO read) continues normally until the frame end of package (EOP) is reached. After this, the following occurs:
    - When Sleep mode is active, all further frames are discarded, flushing the RX FIFO
    - In Hardware Freeze mode, no further frames are delivered to the application and they stay in the receive FIFO.

Note

> The assertion of GRS does not wait for an ongoing FIFO read transaction on the application side of the FIFO (FIFO read).

53.5.9.4.3 Graceful stop interrupt (GRA)

The graceful stopped interrupt (GRA) is asserted for the following conditions:

- In Sleep mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- In Hardware Freeze mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- The MAC transmit datapath is stopped for any other condition (GTS, TFC_PAUSE, pause received).

The GRA interrupt is triggered only once when the stopped state is entered. If the interrupt is cleared while the stop condition persists, no further interrupt is triggered.

53.5.10 IEEE 1588 functions

To allow for IEEE 1588 or similar time synchronization protocol implementations, the MAC is combined with a time-stamping module to support precise time-stamping of incoming and outgoing frames. Set ENET*n*_ECR[EN1588] to enable 1588 support.

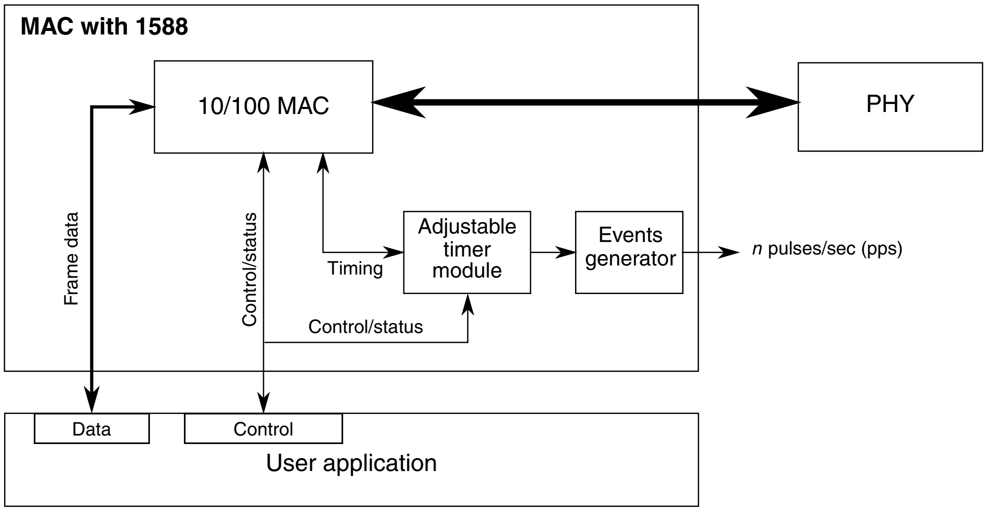


Figure 53-9. IEEE 1588 functions overview

53.5.10.1 Adjustable timer module

The adjustable timer module (TSM) implements the free-running counter (FRC), which generates the timestamps. The FRC operates with the time-stamping clock, which can be set to any value depending on your system requirements.

Through dedicated correction logic, the timer can be adjusted to allow synchronization to a remote master and provide a synchronized timing reference to the local system. The timer can be configured to cause an interrupt after a fixed time period, to allow synchronization of software timers or perform other synchronized system functions.

The timer is typically used to implement a period of one second; hence, its value ranges from 0 to (1 × 10^9)-1. The period event can trigger an interrupt, and software can maintain the seconds and hours time values as necessary.

53.5.10.1.1 Adjustable timer implementation

The adjustable timer consists of a programmable counter/accumulator and a correction counter. The periods of both counters and their increment rates are freely configurable, allowing very fine tuning of the timer.

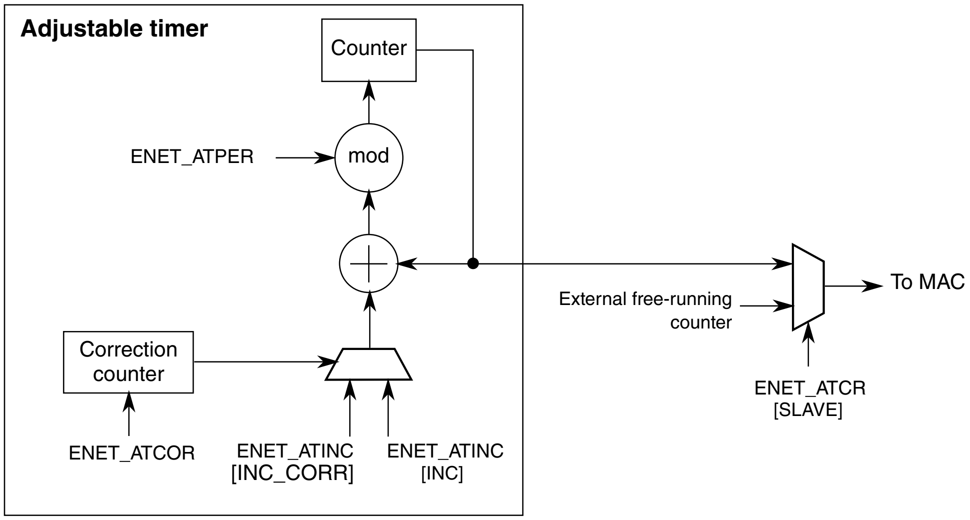


Figure 53-10. Adjustable timer implementation detail

The counter produces the current time. During each time-stamping clock cycle, a constant value is added to the current time as programmed in ENET*n*_ATINC. The value depends on the chosen time-stamping clock frequency. For example, if it operates at 125 MHz, setting the increment to eight represents 8 ns.

The period, configured in ENET*n*_ATPER, defines the modulo when the counter wraps. In a typical implementation, the period is set to 1×10^9 so that the counter wraps every second, and hence all timestamps represent the absolute nanoseconds within the one second period. When the period is reached, the counter wraps to start again respecting the period modulo. This means it does not necessarily start from zero, but instead the counter is loaded with the value (Current + Inc –(1×10^9)), assuming the period is set to 1×10^9.

The correction counter operates fully independently, and increments by one with each time-stamping clock cycle. When it reaches the value configured in ENET*n*_ATCOR, it restarts and instructs the timer once to increment by the correction value, instead of the normal value.

The normal and correction increments are configured in ENET*n*_ATINC. To speed up the timer, set the correction increment more than the normal increment value. To slow down the timer, set the correction increment less than the normal increment value.

The correction counter only defines the distance of the corrective actions, not the amount. This allows very fine corrections and low jitter (in the range of 1 ns) independent of the chosen clock frequency.

By enabling slave mode (ENET*n*_ATCR[SLAVE] = 1), the timer is ignored and the current time is externally provided from one of the external modules. See the Chip Configuration details for which clock source is used. This is useful if multiple modules within the system must operate from a single timer. When slave mode is enabled, you still must set ENET*n*_ATINC[INC] to the value of the master, since it is used for internal comparisons.

53.5.10.2 Transmit timestamping

Only 1588 event frames need to be time-stamped on transmit. The client application (for example, the MAC driver) should detect 1588 event frames and set TxBD[TS] together with the frame.

If TxBD[TS] is set, the MAC records the timestamp for the frame in ENET*n*_ATSTMP. ENET*n*_EIR[TS_AVAIL] is set to indicate that a new timestamp is available.

Software implements a handshaking procedure by setting TxBD[TS] when it transmits the frame for which a timestamp is needed, and then waits for ENET*n*_EIR[TS_AVAIL] to determine when the timestamp is available. The timestamp is then read from ENET*n*_ATSTMP. This is done for all event frames. Other frames do not use TxBD[TS] and, therefore, do not interfere with the timestamp capture.

53.5.10.3 Receive timestamping

When a frame is received, the MAC latches the value of the timer when the frame's start of frame delimiter (SFD) field is detected, and provides the captured timestamp on RxBD[1588 timestamp]. This is done for all received frames.

53.5.10.4 Time synchronization

The adjustable timer module is available to synchronize the local clock of a node to a remote master. It implements a free running 32-bit counter, and also contains an additional correction counter.

The correction counter increases or decreases the rate of the free running counter, enabling very fine granular changes of the timer for synchronization, yet adding only very low jitter when performing corrections.

The application software implements, in a slave scenario, the required control algorithm, setting the correction to compensate for local oscillator drifts and locking the timer to the remote master clock on the network.

The timer and all timestamp-related information should be configured to show the true nanoseconds value of a second (in other words, the timer is configured to have a period of one second). Hence, the values range from 0 to $(1 \times 10^9)–1$. In this application, the seconds counter is implemented in software using an interrupt function that is executed when the nanoseconds counter wraps at 1×10^9.

53.5.10.5 Input Capture and Output Compare

The Input Capture Output Compare block can be used to provide precise hardware timing for input and output events.

53.5.10.5.1 Input capture

The TCCR*n* capture registers latch the time value when the corresponding external event occurs. An event can be a rising-, falling-, or either-edge of one of the 1588_TMR*n* signals. An event will cause the corresponding TCSR*n*[TF] timer flag to be set, indicating that an input capture has occurred. If the corresponding interrupt is enabled with the TCSR*n*[TIE] field, an interrupt can be generated.

53.5.10.5.2 Output compare

The TCCR*n* compare registers are loaded with the time at which the corresponding event should occur. When the ENET free-running counter value matches the output compare reference value in the TCCR*n* register, the corresponding flag, TCSR*n*[TF], is set, indicating that an output compare has occurred. The corresponding interrupt, if enabled by TCSR*n*[TIE], will be generated.The corresponding 1588_TMR*n* output signal will be asserted according to TCSR*n*[TMODE].

53.5.10.5.3 DMA requests

A DMA request can be enabled by setting TCSR*n*[TDRE]. The corresponding DMA request is generated when the TCSR*n*[TF] timer flag is set. When the DMA has completed, the corresponding TCSRn[TF] flag is cleared.

53.5.11 FIFO thresholds

The core FIFO thresholds are fully programmable to dynamically change the FIFO operation.

For example, store and forward transfer can be enabled by a simple change in the FIFO threshold registers.

The thresholds are defined in 64-bit words.

The receive and transmit FIFOs both have a depth of 256 words.

53.5.11.1 Receive FIFO

Four programmable thresholds are available, which can be set to any value to control the core operation.

Table 53-27. Receive FIFO thresholds definition

Register	Description
ENET*n*_RSFL [RX_SECTION_FULL]	When the FIFO level reaches the ENET*n*_RSFL value, the MAC status signal is asserted to indicate that data is available in the receive FIFO (cut-through operation). Once asserted, if the FIFO empties below the threshold set with ENET*n*_RAEM and if the end-of-frame is not yet stored in the FIFO, the status signal is deasserted again. If a frame has a size smaller than the threshold (in other words, an end-of-frame is available for the frame), the status is also asserted. To enable store and forward on the receive path, clear ENET*n*_RSFL. The MAC status signal is asserted only when a complete frame is stored in the receive FIFO. When programming a non-zero value to ENET*n*_RSFL (cut-through operation) it should be greater than ENET*n*_RAEM.
ENET*n*_RAEM [RX_ALMOST_EMPTY]	When the FIFO level reaches the ENET*n*_RAEM value, and the end-of-frame has not been received, the core receive read control stops the FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. Set ENET*n*_RAEM to a minimum of six.
ENET*n*_RAFL [RX_ALMOST_FULL]	When the FIFO level approaches the maximum and there is no more space remaining for at least ENETn_RAFL number of words, the MAC control logic stops writing data in the FIFO and truncates the receive frame to avoid FIFO overflow. The corresponding error status is set when the frame is delivered to the application. Set ENET*n*_RAFL to a minimum of 4.
ENET*n*_RSEM [RX_SECTION_EMPTY]	When the FIFO level reaches the ENET*n*_RSEM value, an indication is sent to the MAC transmit logic, which generates an XOFF pause frame. This indicates FIFO congestion to the remote Ethernet client. When the FIFO level goes below the value programmed in ENET*n*_RSEM, an indication is sent to the MAC transmit logic, which generates an XON pause frame. This indicates the FIFO congestion is cleared to the remote Ethernet client. Clearing ENET*n*_RSEM disables any pause frame generation.

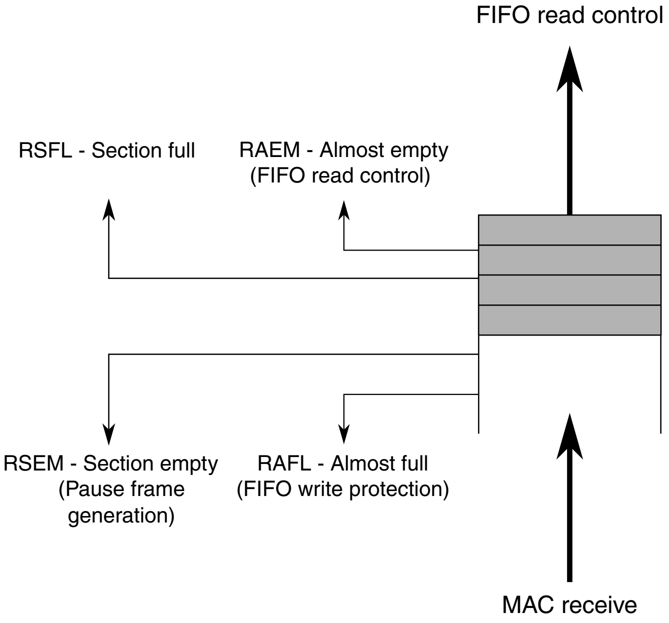


Figure 53-11. Receive FIFO overview

53.5.11.2 Transmit FIFO

Four programmable thresholds are available which control the core operation.

Table 53-28. Transmit FIFO thresholds definition

Register	Description
ENET*n*_TAEM [TX_ALMOST _EMPTY]	When the FIFO level reaches the ENET*n*_TAEM value and no end-of-frame is available for the frame, the MAC transmit logic avoids a FIFO underflow by stopping FIFO reads and transmitting the Ethernet frame with an MII error indication. Set ENET*n*_TAEM to a minimum of 4.
ENET*n*_TAFL [TX_ALMOST _FULL]	When the FIFO level approaches the maximum, so that there is no more space for at least ENET*n*_TAFL number of words, the MAC deasserts its control signal to the application. If the application does not react on this signal, the FIFO write control logic avoids FIFO overflow by truncating the current frame and setting the error status. As a result, the frame is transmitted with an MII error indication. Set ENET*n*_TAFL to a minimum of 4. Larger values allow more latency for the application to react on the MAC control signal deassertion, before the frame is truncated. A typical setting is 8, which offers 3–4 clock cycles of latency to the application to react on the MAC control signal deassertion.
ENET*n*_TSEM [TX_SECTION _EMPTY]	When the FIFO level reaches the ENET*n*_TSEM value, a MAC status signal is deasserted to indicate that the transmit FIFO is getting full. This gives the ENET module an indication to slow or stop its write transaction to avoid a buffer overflow. This is a pure indication function to the application. It has no effect within the MAC. When ENET*n*_TSEM is 0, the signal is never deasserted.
ENET*n*_TFWR	When the FIFO level reaches the ENET*n*_TFWR value and when STRFWD is cleared, the MAC transmit control logic starts frame transmission before the end-of-frame is available in the FIFO (cut-through operation).

Table 53-28. Transmit FIFO thresholds definition

Register	Description
	If a complete frame has a size smaller than the ENET*n*_TFWR threshold, the MAC also transmits the frame to the line. To enable store and forward on the transmit path, set STRFWD. In this case, the MAC starts to transmit data only when a complete frame is stored in the transmit FIFO.

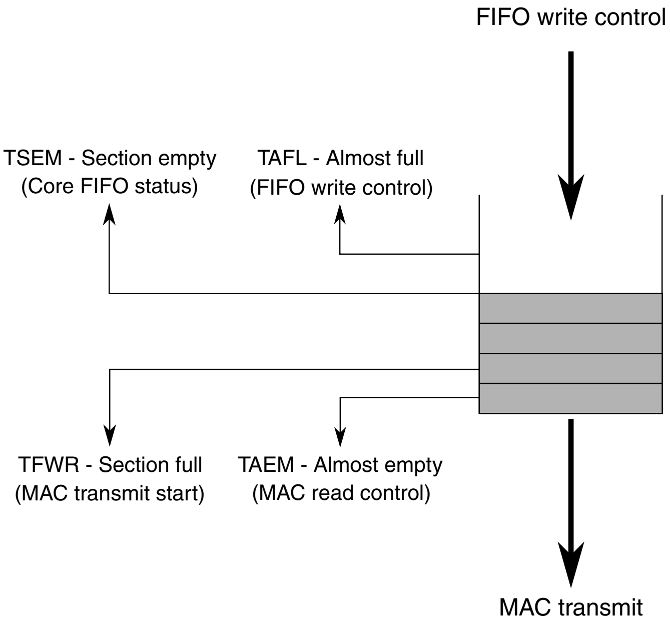


Figure 53-12. Transmit FIFO overview

53.5.12 Loopback options

The core implements external and internal loopback options, which are controlled by the ENET*n*_RCR register fields found here.

Table 53-29. Loopback options

Register field	Description
LOOP	Internal MII loopback. The MAC transmit is returned to the MAC receive. No data is transmitted to the external interfaces. In MII internal loopback, MII_TXCLK and MII_RXCLK must be provided with a clock signal (2.5 MHz for 10 Mbit/s, and 25 MHz for 100 Mbit/s))

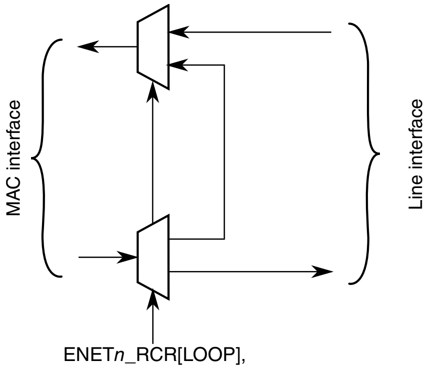


Figure 53-13. Loopback options

53.5.13 Legacy buffer descriptors

To support the Ethernet controller on previous Freescale devices, legacy FEC buffer descriptors are available. To enable legacy support, clear ENET*n*_ECR[1588EN].

NOTE

- The legacy buffer descriptor tables show the byte order for big-endian chips. DBSWP must be set to 0 after reset to enable big-endian mode.

53.5.13.1 Legacy receive buffer descriptor

The following table shows the legacy FEC receive buffer descriptor. Table 53-33 contains the descriptions for each field.

Table 53-30. Legacy FEC receive buffer descriptor (RxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data length															
Offset + 4	Rx data buffer pointer — high halfword															
Offset + 6	Rx data buffer pointer — low halfword															

53.5.13.2 Legacy transmit buffer descriptor

The following table shows the legacy FEC transmit buffer descriptor. Table 53-35 contains the descriptions for each field.

Table 53-31. Legacy FEC transmit buffer descriptor (TxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	TO1	W	TO2	L	TC	ABC[1]	—	—	—	—	—	—	—	—	—
Offset + 2	Data Length															
Offset + 4	Tx Data Buffer Pointer — high halfword															
Offset + 6	Tx Data Buffer Pointer — low halfword															

[1] This field is not supported by the uDMA.

53.5.14 Enhanced buffer descriptors

This section provides a description of the enhanced operation of the driver/uDMA via the buffer descriptors.

It is followed by a detailed description of the receive and transmit descriptor fields. To enable the enhanced features, set ENET*n*_ECR[1588EN].

NOTE

> The enhanced buffer descriptor tables show the byte order for big-endian chips. DBSWP must be set to 0 after reset to enable big-endian mode.

53.5.14.1 Enhanced receive buffer descriptor

The following table shows the enhanced uDMA receive buffer descriptor. Table 53-33 contains the descriptions for each field.

Table 53-32. Enhanced uDMA receive buffer descriptor (RxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data length															
Offset + 4	Rx data buffer pointer – high halfword															
Offset + 6	Rx data buffer pointer – low halfword															
Offset + 8	ME	—	—	—	—	PE	CE	UC	INT	—	—	—	—	—	—	—

Table continues on the next page...

Table 53-32. Enhanced uDMA receive buffer descriptor (RxBD) (continued)

Offset + A	VPCP			—	—	—	—	—	—	—	ICE	PCR	—	VLAN	IPV6	FRAG
Offset + C	Header length					—	—	—	Protocol type							
Offset + E	Payload checksum															
Offset + 10	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – high halfword															
Offset + 16	1588 timestamp – low halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 53-33. Receive buffer descriptor field definitions

Word	Field	Description
Offset + 0	0 E	Empty. Written by the MAC (= 0) and user (= 1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	1 RO1	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 0	2 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in ENET*n*_RDSR.
Offset + 0	3 RO2	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 0	4 L	Last in frame. Written by the uDMA. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	5–6	Reserved, must be cleared.
Offset + 0	7 M	Miss. Written by the MAC. This field is set by the MAC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use this field to quickly determine whether the frame was destined to this station. This field is valid only if the L and PROM fields are set to 1. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode. The information needed for this field comes from the promiscuous_miss(ff_rx_err_stat[26]) sideband signal.
Offset + 0	8 BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).

Table continues on the next page...

Table 53-33. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + 0	9 MC	Set if the DA is multicast and not BC.
Offset + 0	10 LG	Receive frame length violation. Written by the MAC. A frame length greater than RCR[MAX_FL] was recognized. This field is valid only if the L field is set. The receive data is not altered in any way unless the length exceeds TRUNC_FL bytes.
Offset + 0	11 NO	Receive non-octet aligned frame. Written by the MAC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error or a PHY error occurred. This field is valid only if the L field is set. If this field is set, the CR field is not set.
Offset + 0	12	Reserved, must be cleared.
Offset + 0	13 CR	Receive CRC or frame error. Written by the MAC. This frame contains a PHY or CRC error and is an integral number of octets in length. This field is valid only if the L field is set.
Offset + 0	14 OV	Overrun. Written by the MAC. A receive FIFO overrun occurred during frame reception. If this field is set, the other status fields, M, LG, NO, and CR, lose their normal meaning and are zero. This field is valid only if the L field is set.
Offset + 0	15 TR	Set if the receive frame is truncated (frame length >TRUNC_FL). If the TR field is set, the frame must be discarded and the other error fields must be ignored because they may be incorrect.
Offset + 2	0–15 Data Length	Data length. Written by the MAC. Data length is the number of octets written by the MAC into this BD's data buffer if L is cleared (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the MAC once as the BD is closed.
0ffset + 4	0–15 Data buffer pointer high	Receive data buffer pointer, high halfword[1]
Offset + 6	0–15 Data buffer pointer low	Receive data buffer pointer, low halfword
Offset + 8	0 ME	MAC error. This field is written by the uDMA. This field means that the frame stored in the system memory was received with an error (typically, a receive FIFO overflow). This field is only valid when the L field is set.
Offset + 8	1–4	Reserved, must be cleared.
Offset + 8	5 PE	PHY Error. This field is written by the uDMA. Set to "1"when the frame was received with an Error character on the PHY interface. The frame is invalid. This field is valid only when the L field is set.
Offset + 8	6 CE	Collision. This field is written by the uDMA. Set when the frame was received with a collision detected during reception. The frame is invalid and sent to the user application. This field is valid only when the L field is set.
Offset + 8	7 UC	Unicast. This field is written by the uDMA, and means that the frame is unicast. This field is valid regardless of whether the L field is set.
Offset + 8	8 INT	Generate RXB/RXF interrupt. This field is set by the user to indicate that the uDMA is to generate an interrupt on the *dma_int_rxb* / *dma_int_rxfevent*.
Offset + 8	9–15	Reserved, must be cleared.

Table continues on the next page...

Table 53-33. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + A	0–2 VPCP	VLAN priority code point. This field is written by the uDMA to indicate the frame priority level. Valid values are from 0 (best effort) to 7 (highest). This value can be used to prioritize different classes of traffic (e.g., voice, video, data). This field is only valid if the L field is set.
Offset + A	3–9	Reserved, must be cleared.
Offset + A	10 ICE	IP header checksum error. This is an accelerator option. This field is written by the uDMA. Set when either a non-IP frame is received or the IP header checksum was invalid. An IP frame with less than 3 bytes of payload is considered to be an invalid IP frame. This field is only valid if the L field is set.
Offset + A	11 PCR	Protocol checksum error. This is an accelerator option. This field is written by the uDMA. Set when the checksum of the protocol is invalid or an unknown protocol is found and checksumming could not be performed. This field is only valid if the L field is set.
Offset + A	12	Reserved, must be cleared.
Offset + A	13 VLAN	VLAN. This is an accelerator option. This field is written by the uDMA. It means that the frame has a VLAN tag. This field is valid only if the L field is set.
Offset + A	14 IPV6	IPV6 Frame. This field is written by the uDMA. This field indicates that the frame has an IPv6 frame type. If this field is not set it means that an IPv4 or other protocol frame was received. This field is valid only if the L field is set.
Offset + A	15 FRAG	IPv4 Fragment.This is an accelerator option.This field is written by the uDMA. It indicates that the frame is an IPv4 fragment frame. This field is only valid when the L field is set.
Offset + C	0–4 Header length	Header length. This is an accelerator option. This field is written by the uDMA. This field is the sum of 32-bit words found within the IP and its following protocol headers. If an IP datagram with an unknown protocol is found, then the value is the length of the IP header. If no IP frame or an erroneous IP header is found, the value is 0. The following values are minimum values if no header options exist in the respective headers: • ICMP/IP: 6 (5 IP header, 1 ICMP header) • UDP/IP: 7 (5 IP header, 2 UDP header) • TCP/IP: 10 (5 IP header, 5 TCP header) This field is only valid if the L field is set.
Offset + C	5–7	Reserved, must be cleared.
Offset + C	8–15 Protocol type	Protocol type. This is an accelerator option. The 8-bit protocol field found within the IP header of the frame. It is valid only when ICE is cleared. This field is valid only when the L field is set.
Offset + E	0–15 Payload checksum	Internet payload checksum. This is an accelerator option. It is the one's complement sum of the payload section of the IP frame. The sum is calculated over all data following the IP header until the end of the IP payload. This field is valid only when the L field is set.
Offset + 10	0 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 10	1–15	Reserved, must be cleared.
Offset + 12	0–15	Reserved, must be cleared.
Offset + 14	0–15 1588 timestamp	This value is written by the uDMA. It is only valid if the L field is set.
Offset + 16		
Offset + 18	0–15	Reserved, must be cleared.

Table 53-33. Receive buffer descriptor field definitions

Word	Field	Description
– Offset + 1E		

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the MAC. The Ethernet controller never modifies this value.

53.5.14.2 Enhanced transmit buffer descriptor

Table 53-34. Enhanced transmit buffer descriptor (TxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	TO1	W	TO2	L	TC	—	—	—	—	—	—	—	—	—	—
Offset + 2	Data length															
Offset + 4	Tx Data Buffer Pointer – high halfword															
Offset + 6	Tx Data Buffer Pointer – low halfword															
Offset + 8	—	INT	TS	PINS	IINS	—	—	—	—	—	—	—	—	—	—	—
Offset + A	TXE	—	UE	EE	FE	LCE	OE	TSE	—	—	—	—	—	—	—	—
Offset + C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 10	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – high halfword															
Offset + 16	1588 timestamp – low halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 53-35. Enhanced transmit buffer descriptor field definitions

Word	Field	Description
Offset + 0	0 R	Ready. Written by the MAC and user. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The MAC clears this field after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this field is set.
Offset + 0	1	Transmit software ownership. This field is reserved for software use. This read/ write field is not modified by hardware and its value does not affect hardware.

Table continues on the next page...

Table 53-35. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
	TO1	
Offset + 0	2 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	3 TO2	Transmit software ownership. This field is reserved for use by software. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 0	4 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 0	5 TC	Transmit CRC. Written by user, and valid only when L is set. 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte This field is valid only when the L field is set.
Offset + 0	6 ABC	Append bad CRC. **Note:** This field is not supported by the uDMA and is ignored.
Offset + 0	7–15	Reserved, must be cleared.
Offset + 2	0–15 Data Length	Data length, written by user. Data length is the number of octets the MAC should transmit from this BD's data buffer. It is never modified by the MAC.
Offset + 4	0–15 Data buffer pointer high	Tx data buffer pointer, high halfword. The transmit buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 8. The buffer must reside in memory external to the MAC. This value is never modified by the Ethernet controller.
Offset + 6	0–15 Data buffer pointer low	Tx data buffer pointer, low halfword
Offset + 8	0	Reserved, must be cleared.
Offset + 8	1 INT	Generate interrupt flags. This field is written by the user. This field is valid regardless of the L field and must be the same for all EBD for a given frame. The uDMA does not update this value.
Offset + 8	2 TS	Timestamp. This field is written by the user. This indicates that the uDMA is to generate a timestamp frame to the MAC. This field is valid regardless of the L field and must be the same for all EBD for the given frame. The uDMA does not update this value.
Offset + 8	3 PINS	Insert protocol specific checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the protocol checksum and overwrites the corresponding checksum field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + 8	4 IINS	Insert IP header checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the IP header checksum and overwrites the corresponding header field with the calculated value. The checksum field must be cleared by

Table continues on the next page...

Table 53-35. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
		the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + 8	5–15	Reserved, must be cleared.
Offset + A	0 TXE	Transmit error occurred. This field is written by the uDMA. This field indicates that there was a transmit error of some sort reported with the frame. Effectively this field is an OR of the other error fields including UE, EE, FE, LCE, OE, and TSE. This field is valid only when the L field is set.
Offset + A	1	Reserved, must be cleared.
Offset + A	2 UE	Underflow error. This field is written by the uDMA. This field indicates that the MAC reported an underflow error on transmit. This field is valid only when the L field is set.
Offset + A	3 EE	Excess Collision error. This field is written by the uDMA. This field indicates that the MAC reported an excess collision error on transmit. This field is valid only when the L field is set.
Offset + A	4 FE	Frame with error. This field is written by the uDMA. This field indicates that the MAC reported that the uDMA reported an error when providing the packet. This field is valid only when the L field is set.
Offset + A	5 LCE	Late collision error. This field is written by the uDMA. This field indicates that the MAC reported that there was a Late Collision on transmit. This field is valid only when the L field is set.
Offset + A	6 OE	Overflow error. This field is written by the uDMA. This field indicates that the MAC reported that there was a FIFO overflow condition on transmit. This field is only valid when the L field is set.
Offset + A	7 TSE	Timestamp error. This field is written by the uDMA. This field indicates that the MAC reported a different frame type then a timestamp frame. This field is valid only when the L field is set.
Offset + A	8–15	Reserved, must be cleared.
Offset + C	0–15	Reserved, must be cleared.
Offset + E	0–15	Reserved, must be cleared.
Offset + 10	0 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 10	1–15	Reserved, must be cleared.
Offset + 12	0–15	Reserved, must be cleared.
Offset + 14	0–15 1588 timestamp	This value is written by the uDMA . It is valid only when the L field is set.
Offset + 16		
Offset + 18–Offset + 1E	0–15	Reserved, must be cleared.

53.5.15 Client FIFO application interface

The FIFO interface is completely asynchronous from the Ethernet line, and the transmit and receive interface can operate at a different clock rate.

All transfers to/from the user application are handled independently of the core operation, and the core provides a simple interface to user applications based on a two-signal handshake.

53.5.15.1 Data structure description

The data structure defined in the following tables for the FIFO interface must be respected to ensure proper data transmission on the Ethernet line. Byte 0 is sent to and received from the line first.

Table 53-36. FIFO interface data structure

	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Word 0	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Word 1	Byte 15	Byte 14	Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8
...	...							

The size of a frame on the FIFO interface may not be a modulo of 64-bit.

The user application may not care about the Ethernet frame formats in full detail. It needs to provide and receive an Ethernet frame with the following structure:

- Ethernet MAC destination address
- Ethernet MAC source address
- Optional 802.1q VLAN tag (VLAN type and info field)
- Ethernet length/type field
- Payload

Frames on the FIFO interface do not contain preamble and SFD fields, which are inserted and discarded by the MAC on transmit and receive, respectively.

- On receive, CRC and frame padding can be stripped or passed through transparently.
- On transmit, padding and CRC can be provided by the user application, or appended automatically by the MAC independently for each frame. No size restrictions apply.

Note

On transmit, if ENET*n*_TCR[ADDINS] is set, bytes 6–11 of each frame can be set to any value, since the MAC overwrites the bytes with the MAC address programmed in the ENET*n*_PAUR and ENET*n*_PALR registers.

Table 53-37. FIFO interface frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–13	Length/type field
14–N	Payload data

VLAN-tagged frames are supported on both transmit and receive, and implement additional information (VLAN type and info).

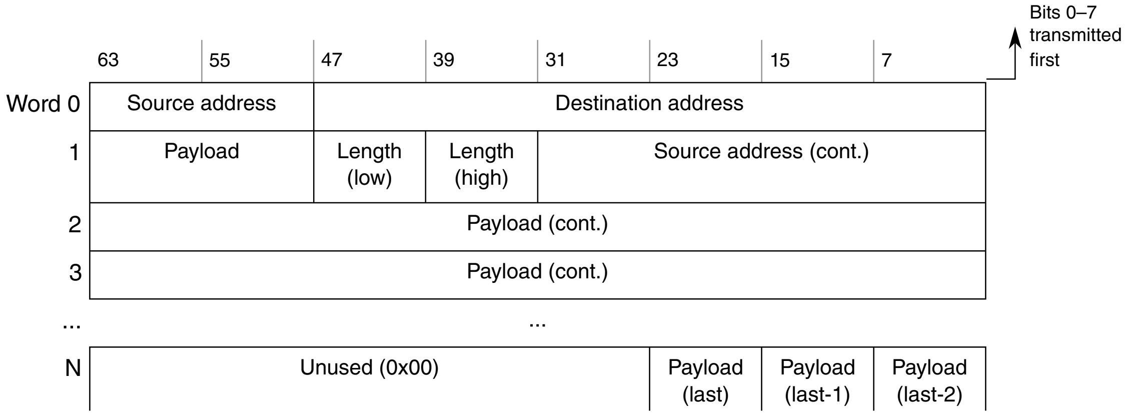
Table 53-38. FIFO interface VLAN frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–15	VLAN tag and info
16–17	Length/type field
18–N	Payload data

Note

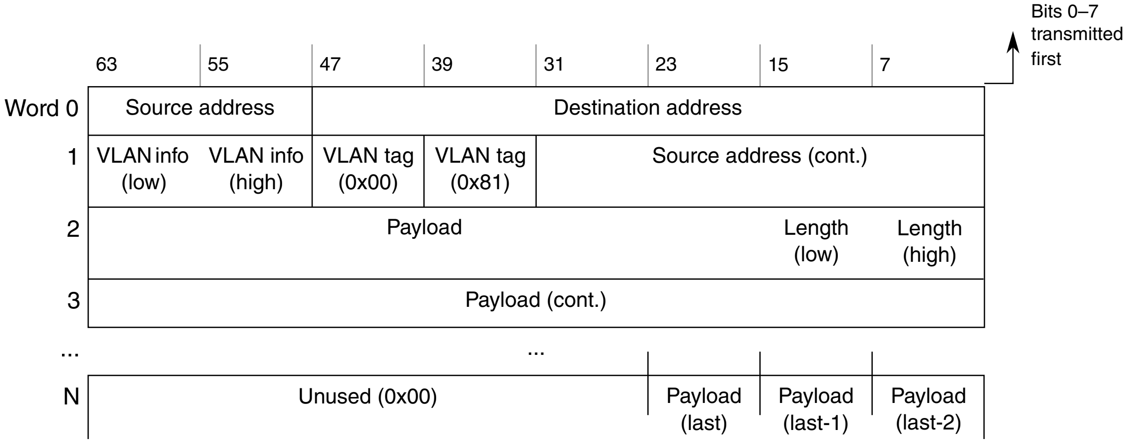
The standard defines that the LSB of the MAC address is sent/ received first, while for all the other header fields — in other words, length/type, VLAN tag, VLAN info, and pause quanta — the MSB is sent/received first.

53.5.15.2 Data structure examples



	63	55	47	39	31	23	15	7
Word 0	Source address		Destination address					
1	Payload		Length (low)	Length (high)	Source address (cont.)			
2	Payload (cont.)							
3	Payload (cont.)							
...	...							
N	Unused (0x00)					Payload (last)	Payload (last-1)	Payload (last-2)

Figure 53-14. Normal Ethernet frame 64-bit mapping example



	63	55	47	39	31	23	15	7
Word 0	Source address		Destination address					
1	VLAN info (low)	VLAN info (high)	VLAN tag (0x00)	VLAN tag (0x81)	Source address (cont.)			
2	Payload						Length (low)	Length (high)
3	Payload (cont.)							
...	...							
N	Unused (0x00)					Payload (last)	Payload (last-1)	Payload (last-2)

Figure 53-15. VLAN-tagged frame 64-bit mapping example

If CRC forwarding is enabled (CRCFWD = 0), the last four valid octets of the frame contain the FCS field. The non-significant bytes of the last word can have any value.

53.5.15.3 Frame status

A MAC layer status word and an accelerator status word is available in the receive buffer descriptor.

The status is available with each frame with the last data of the frame.

If the frame status contains a MAC layer error (for example, CRC or length error), RxBD[ME] is also set with the last data of the frame.

53.5.16 FIFO protection

53.5.16.1 Transmit FIFO underflow

During a frame transfer, when the transmit FIFO reaches the almost empty threshold with no end-of-frame indication stored in the FIFO, the MAC logic:

- Stops reading data from the FIFO
- Asserts the MII error signal (MII_TXER) (1 in Figure 53-16) to indicate that the fragment already transferred is not valid
- Deasserts the MII transmit enable signal (MII_TXEN) to terminate the frame transfer (2)

After an underflow, when the application completes the frame transfer (3), the MAC transmit logic discards any new data available in the FIFO until the end of packet is reached (4) and sets the enhanced TxBD[UE] field.

The MAC starts to transfer data on the MII interface when the application sends a new frame with a start of frame indication (5).

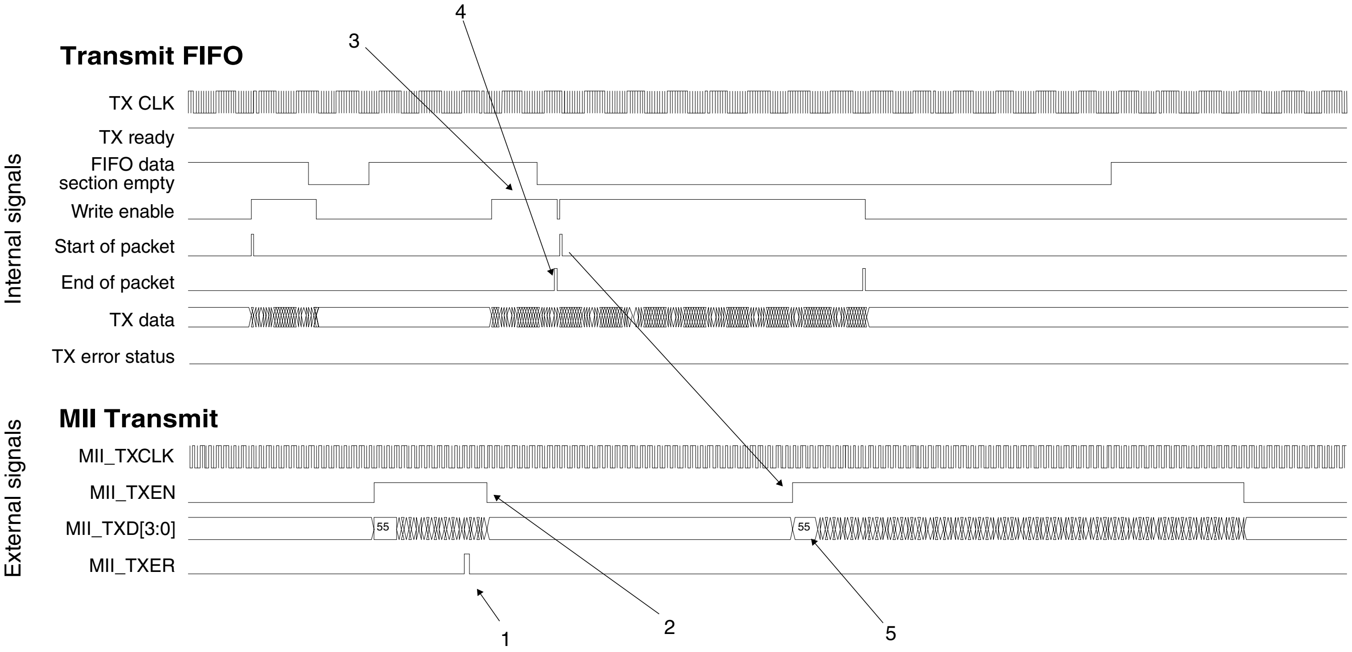


Figure 53-16. Transmit FIFO underflow protection

53.5.16.2 Transmit FIFO overflow

On the transmit path, when the FIFO reaches the programmable almost full threshold, the internal MAC ready signal is deasserted. The application should stop sending new data.

However, if the application keeps sending data, the transmit FIFO overflows, corrupting contents that were previously stored. The core logic sets the enhanced TxBD[OE] field for the next frame transmitted to indicate this overflow occurence.

Note

> Overflow is a fatal error and must be addressed by resetting the core or clearing ENET*n*_ECR[ETHER_EN], to clear the FIFOs and prepare for normal operation again.

53.5.16.3 Receive FIFO overflow

During a frame reception, if the client application is not able to receive data (1), the MAC receive control truncates the incoming frame when the FIFO reaches the programmable almost-full threshold to avoid an overflow.

The frame is subsequently received on the FIFO interface with an error indication (enhanced RxBD[ME] field set together with receive end-of-packet) (2) with the truncation error status field set (3).

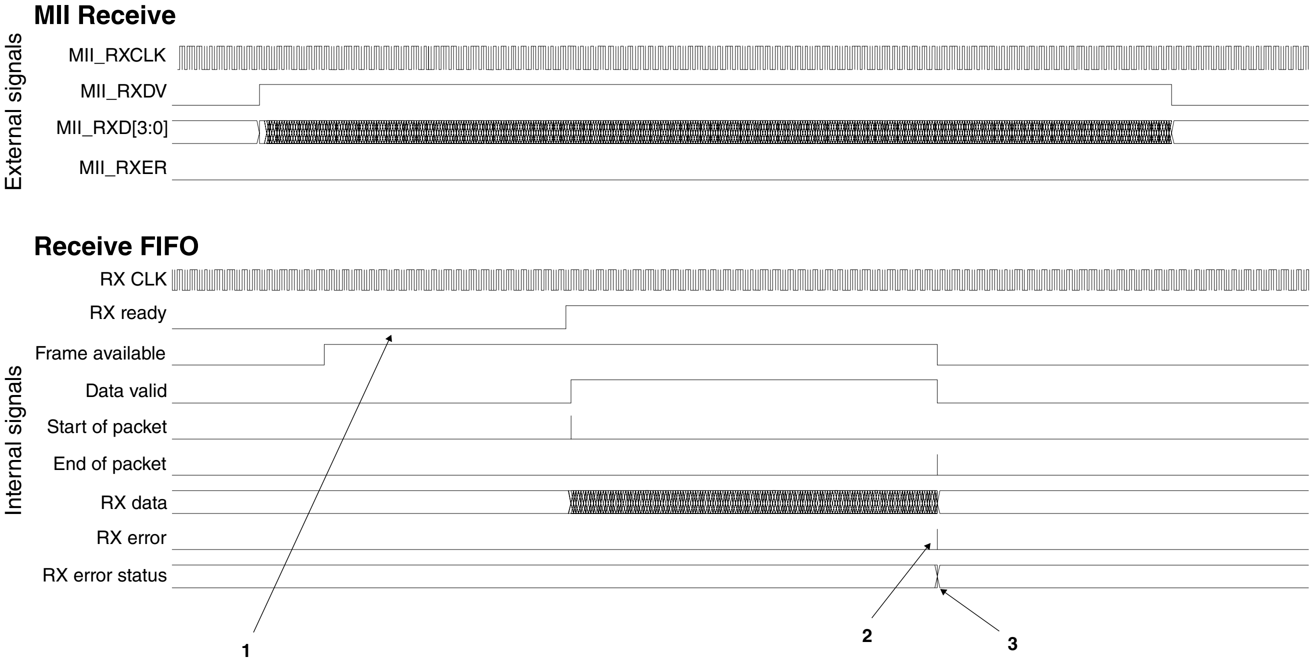


Figure 53-17. Receive FIFO overflow protection

53.5.17 Reference clock

The input clocks to the ENET module must meet the specifications in the following table.

Ethernet speed mode	Ethernet bus clock	Minimum ENET system clock
10 Mbit/s	2.5 MHz	5 MHz
100 Mbit/s	25.0 MHz	50 MHz

53.5.18 PHY management interface

The MDIO interface is a two-wire management interface. The MDIO management interface implements a standardized method to access the PHY device management registers.

The core implements a master MDIO interface, which can be connected to up to 32 PHY devices.

53.5.18.1 MDIO clause 22 frame format

The core MDIO master controller communicates with the slave (PHY device) using frames that are defined in the following table.

A complete frame has a length of 64 bits made up of an optional 32-bit preamble, 14-bit command, 2-bit bus direction change, and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock (MDC signal). The MDIO data signal is tri-stated between frames.

The core PHY management interface supports the standard MDIO specification (IEEE 802.3 Clause 22).

Table 53-39. MDIO clause 22 frame structure

ST	OP	PHYADR	REGADR	TA	DATA

Table 53-40. MDIO frame field descriptions

Field	Description
ST (2 bits)	Start indication field, programmed with ENET*n*_MMFR[ST] and equal to 01 for Standard MDIO (Clause 22).
OP (2 bits)	Opcode defines type of operation. Programmed with ENET*n*_MMFR[OP]. 01 Write operation 10 Read operation
PHYADR (5 bits)	Five-bit PHY device address, programmed with ENET*n*_MMFR[PA]. Up to 32 devices can be addressed.

Table continues on the next page...

Table 53-40. MDIO frame field descriptions (continued)

Field	Description
REGADR (5 bits)	Five-bit register address, programmed with ENET*n*_MMFR[RA]. Each PHY can implement up to 32 registers.
TA (2 bits)	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
Data (16 bits)	Data, set by ENET*n*_MMFR[DATA]. Written to or read from the PHY

53.5.18.2 MDIO clause 45 frame format

The extended MDIO frame structure defined in IEEE 802.3 Clause 45 introduces indirect addressing. First, a write transaction to an address register is done, followed by a write or read transaction which will put the 16-bit data in the register or retrieve the register contents respectively. A preamble of 32 bits of logical ones is sent prior to every transaction. The MDIO data signal is tri-stated between frames.

The extended MDIO defines four transactions, which are determined by the two-bit opcode field.

Table 53-41. MDIO clause 45 frame structure

ST	OP	PRTAD	DEVAD	TA	ADDR/DATA

All bits are transmitted from left to right (Preamble bits first) and all fields have their Most-Significant bit sent first (leftmost in above table). The complete frame has a length of 64 bits (32-bit preamble, 14-bit command, 2-bit bus direction change, 16-bit data). Each bit is transferred with the rising edge of the MDIO clock (MDC).

The fields and transactions are summarized in the following tables.

Table 53-42. MDIO clause 45 frame field descriptions

Field	Description
ST	Start indication. Indicates the end of the preamble and start of the frame. This value is 00 for extended MDIO (Clause 45) frames.
OP	Opcode defines if a read or write operation is performed and is programmed with ENET*n*_MMFR[OP]. See Table 53-43 for more information. 00 Address write 01 Write operation 10 Read inc. operation

Table continues on the next page...

Table 53-42. MDIO clause 45 frame field descriptions (continued)

Field	Description
	11 Read operation
PRTAD	The port address specifies a MDIO port. Each Port can have up to 32 devices which each can have a separate set of registers.
DEVAD	Device address. Up to 32 devices can be addressed (within a port).
TA	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
ADDR/DATA	16-bit address (for address write) or data, set by ENET*n*_MMFR[DATA], written to or read from the PHY.

Table 53-43. MDIO Clause 45 Transactions

Transaction Type	Description
Address	A write transaction to the internal address register of the device/port. The data section of the frame contains the value to be stored in the device's internal address "pointer" register for further transactions.
Write	Data write to a register. The 16 bit data will be written to the register identified by the device-internal address.
Read	Data is read from the register identified by the device-internal address.
Read inc.	Read with address postincrement. The register identified by the device-internal address is read. After this, the device-internal address is incremented. If the address register is all '1' (0xFFFF) no increment is done (i.e. increment does not wrap around).

53.5.18.3 MDIO clock generation

The MDC clock is generated from the internal bus clock (i.e., IPS bus clock) divided by the value programmed in ENET*n*_MSCR[MII_SPEED].

53.5.18.4 MDIO operation

To perform an MDIO access, set the MDIO command register (ENET*n*_MMFR) according to the description provided in MII Management Frame Register (ENETn_MMFR).

To check when the programmed access completes, read the ENET*n*_EIR[MII] field.

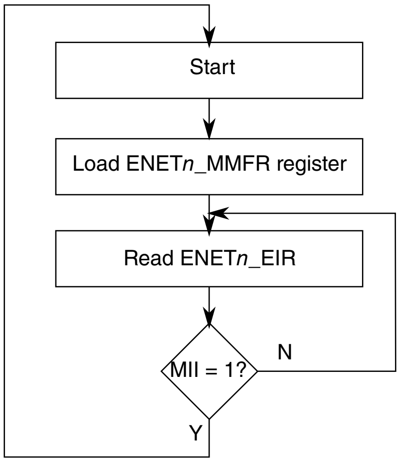


Figure 53-18. MDIO access overview

53.5.19 Ethernet interfaces

The following Ethernet interfaces are implemented:

- Fast Ethernet MII (Media Independent Interface)
- RMII 10/100 using interface converters/gaskets

The following table shows how to configure ENET registers to select each interface.

Mode	ECR[SLEEP]	RCR[RMII_10T]	RCR[RMII_MODE]
MII - 10 Mbit/s	0	—	0
MII - 100 Mbit/s	0	—	0
RMII - 10 Mbit/s	0	1	1
RMII - 100 Mbit/s	0	0	1

53.5.19.1 RMII interface

In RMII receive mode, for normal reception following assertion of CRS_DV, RXD[1:0] is 00 until the receiver determines that the receive event has a proper start-of-stream delimiter (SSD).

The preamble appears (RXD[1:0]=01) and the MACs begin capturing data following detection of SFD.

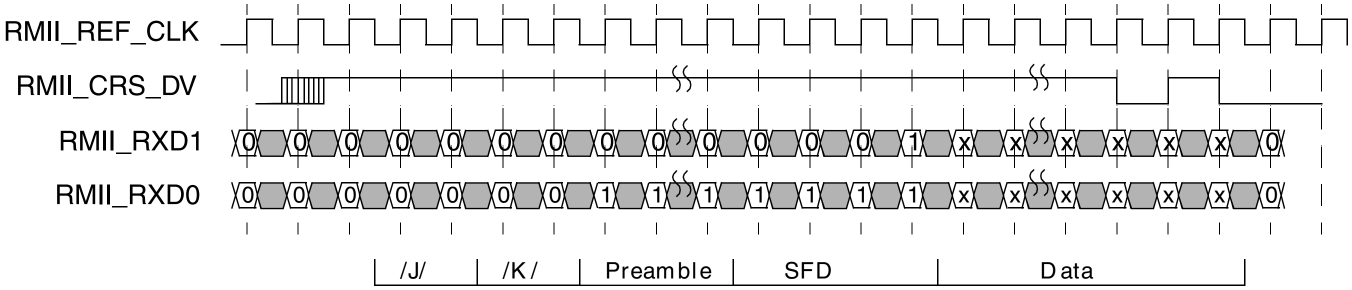


Figure 53-19. RMII receive operation

If a false carrier is detected (bad SSD), then RXD[1:0] is 10 until the end of the receive event. This is a unique pattern since a false carrier can only occur at the beginning of a packet where the preamble is decoded (RXD[1:0] = 01).

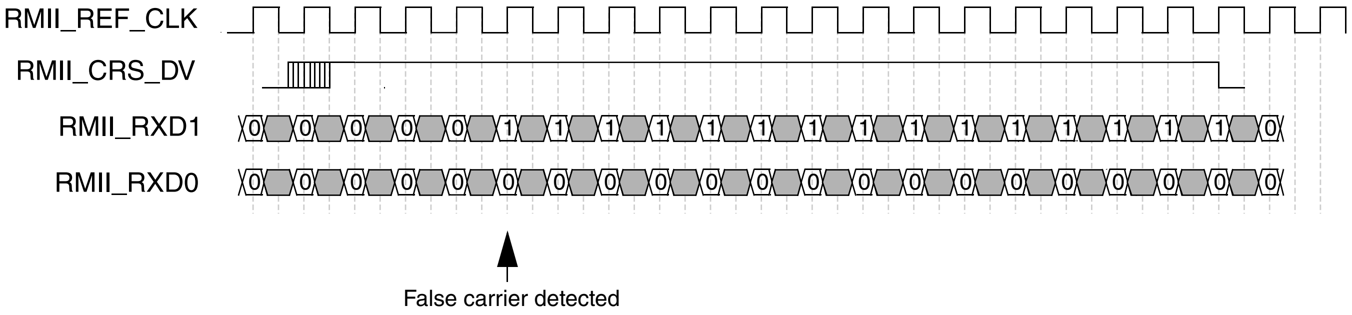


Figure 53-20. RMII receive operation with false carrier

In RMII transmit mode, TXD[1:0] provides valid data for each REF_CLK period while TXEN is asserted.

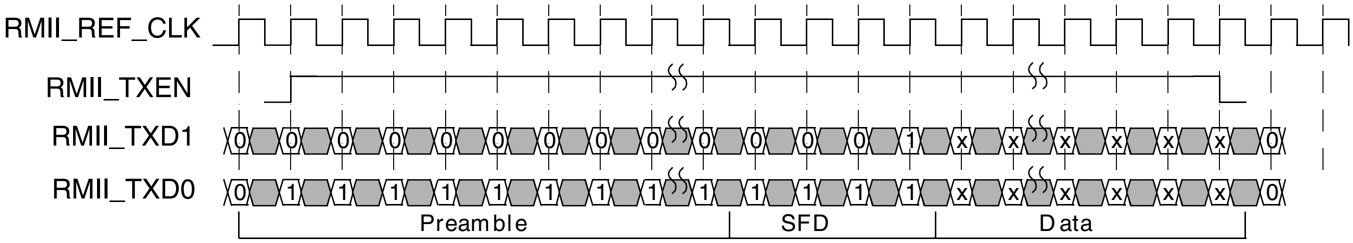


Figure 53-21. RMII transmit operation

53.5.19.2 MII Interface — transmit

On transmit, all data transfers are synchronous to MII_TXCLK rising edge. The MII data enable signal MII_TXEN is asserted to indicate the start of a new frame, and remains asserted until the last byte of the frame is present on the MII_TXD[3:0] bus.

Between frames, MII_TXEN remains deasserted.

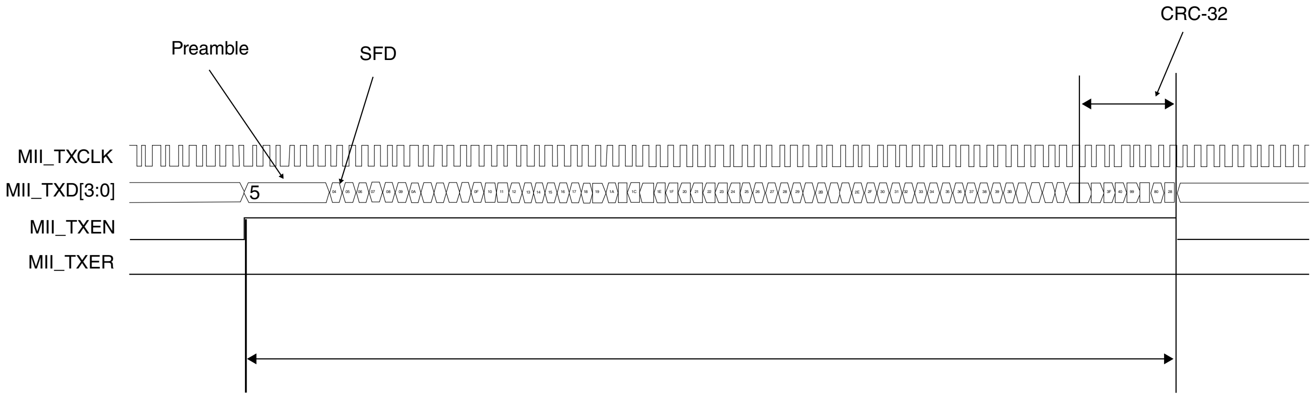


Figure 53-22. MII transmit operation

If a frame is received on the FIFO interface with an error (for example, RxBD[ME] set) the frame is subsequently transmitted with the MII_TXER error signal for one clock cycle at any time during the packet transfer.

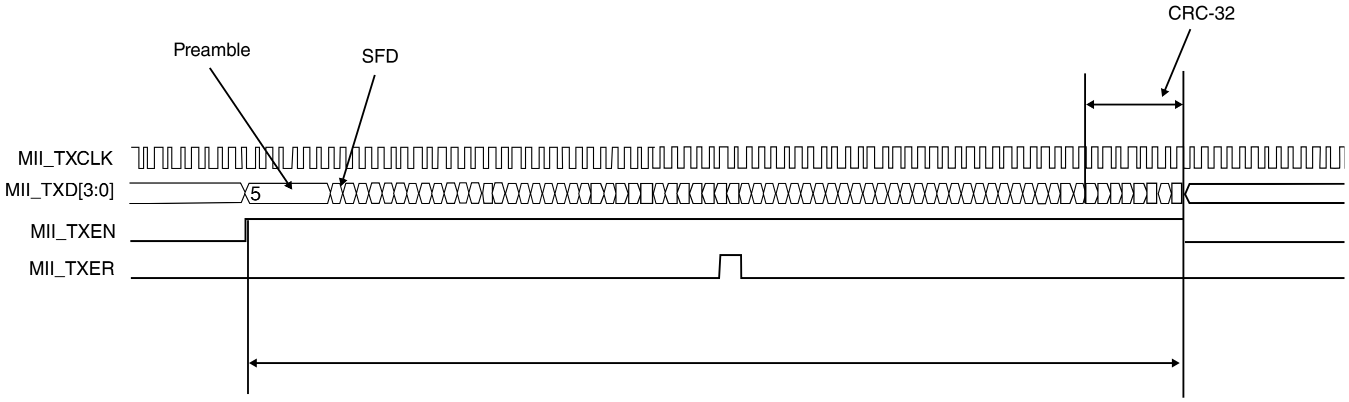


Figure 53-23. MII transmit operation — errored frame

53.5.19.2.1 Transmit with collision — half-duplex

When a collision is detected during a frame transmission (MII_COL asserted), the MAC stops the current transmission, sends a 32-bit jam pattern, and re-transmits the current frame.

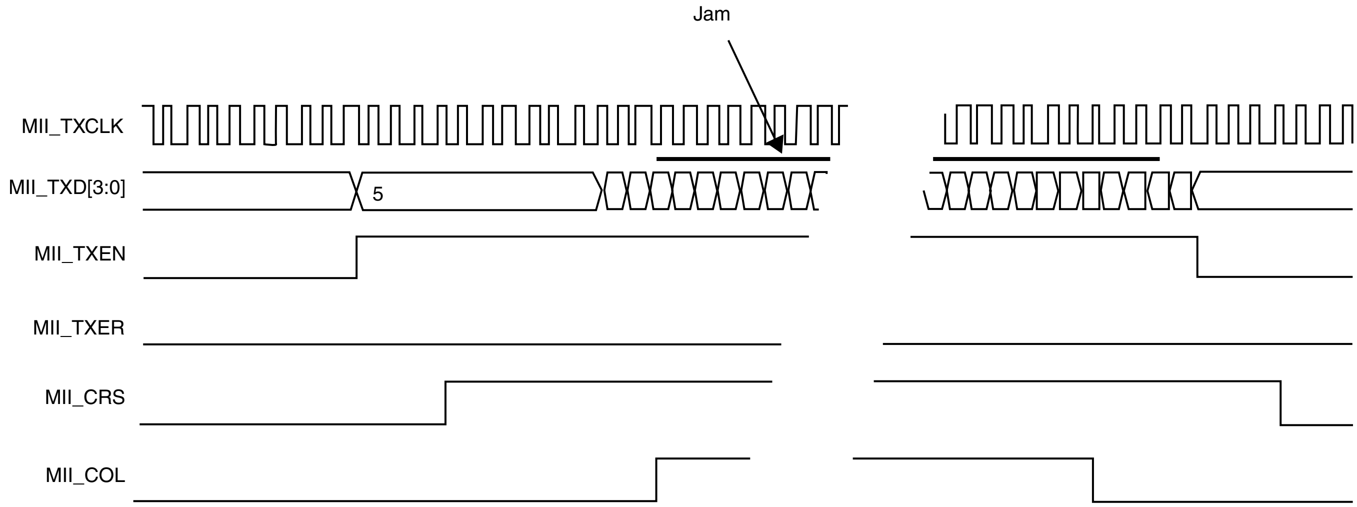


Figure 53-24. MII transmit operation — transmission with collision

53.5.19.3 MII interface — receive

On receive, all signals are sampled on the MII_RXCLK rising edge. The MII data enable signal, MII_RXDV, is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on MII_RXD[3:0] bus.

Between frames, MII_RXDV remains deasserted.

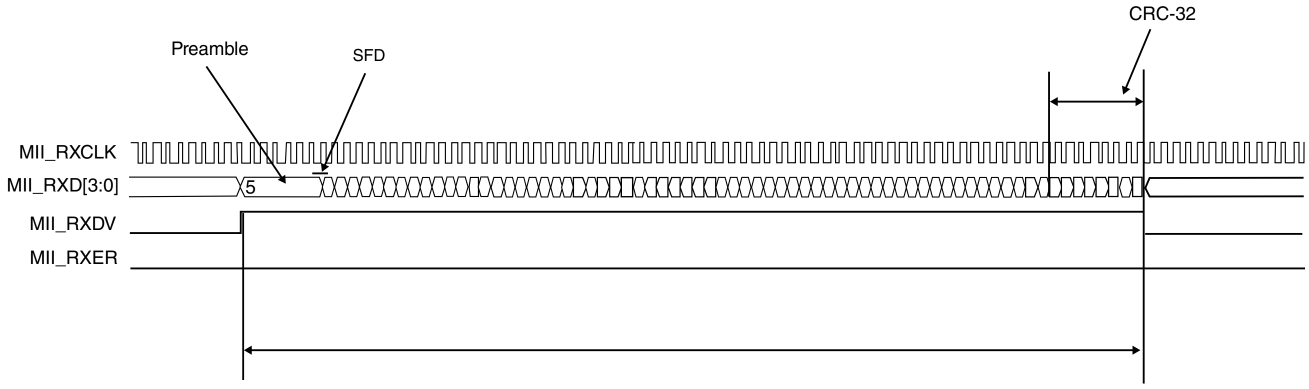


Figure 53-25. MII receive operation

If the PHY detects an error on the frame received from the line, the PHY asserts the MII error signal, MII_RXER, for at least one clock cycle at any time during the packet transfer.

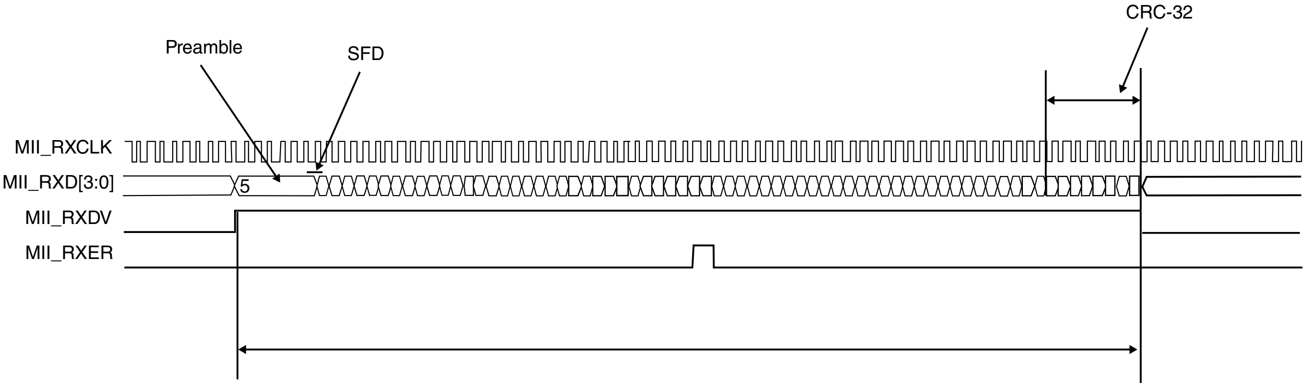


Figure 53-26. MII receive operation — errored frame

A frame received on the MII interface with a PHY error indication is subsequently transferred on the FIFO interface with RxBD[ME] set.

Chapter 54
Reset Generation Module (MC_RGM)

54.1 Introduction

54.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

The following figure shows the MC_RGM block diagram.

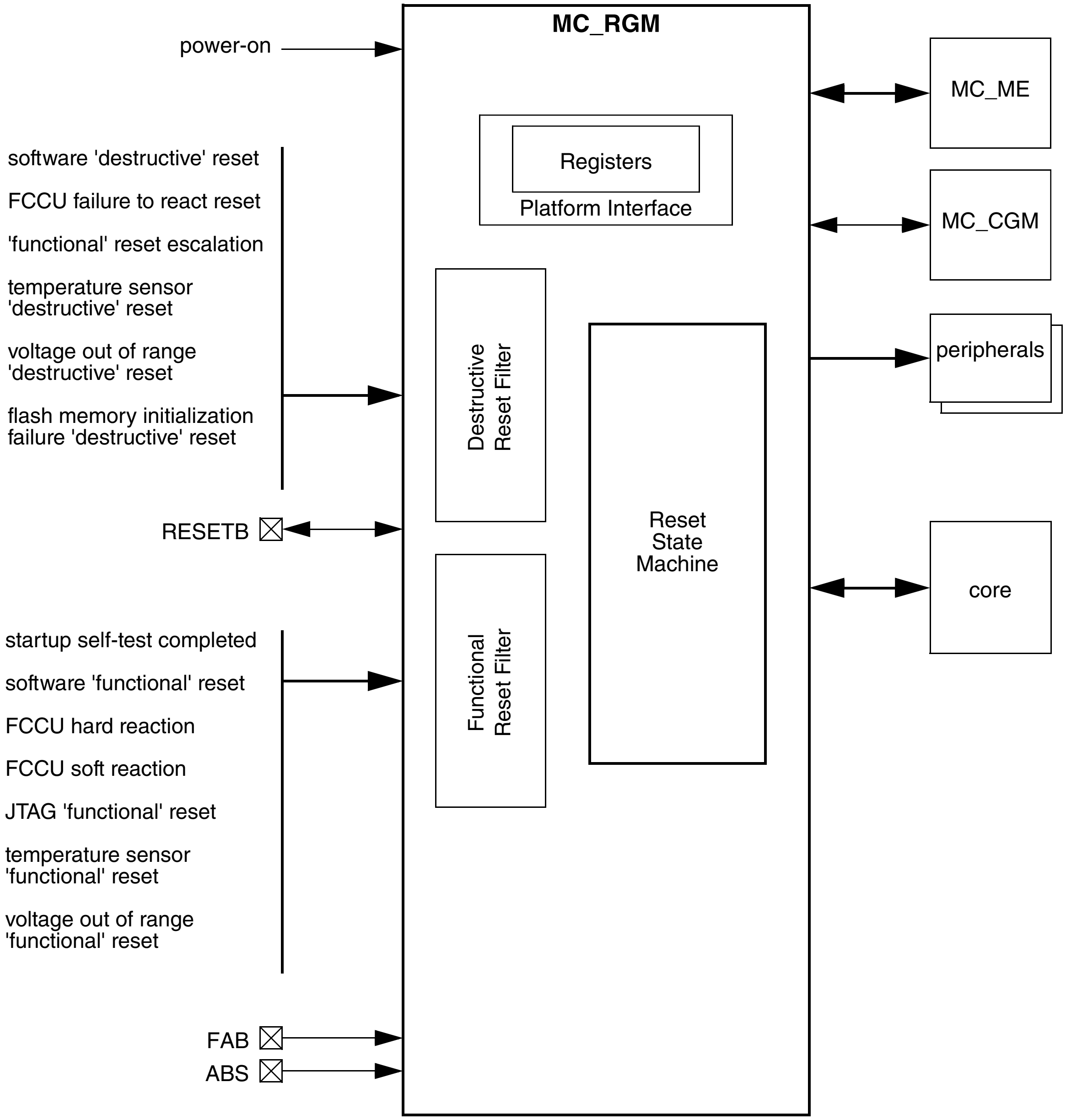

MC_RGM
power-on
software 'destructive' reset
FCCU failure to react reset
'functional' reset escalation
temperature sensor 'destructive' reset
voltage out of range 'destructive' reset
flash memory initialization failure 'destructive' reset
RESETB
startup self-test completed
software 'functional' reset
FCCU hard reaction
FCCU soft reaction
JTAG 'functional' reset
temperature sensor 'functional' reset
voltage out of range 'functional' reset
FAB
ABS
Registers
Platform Interface
Destructive Reset Filter
Functional Reset Filter
Reset State Machine
MC_ME
MC_CGM
peripherals
core

Figure 54-1. MC_RGM Block Diagram

54.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'destructive' resets management
- 'functional' resets management
- signalling of reset events after each reset sequence (reset status flags)
- conversion of reset events to SAFE mode or interrupt request events

- short reset sequence configuration
- bidirectional reset behavior configuration
- boot mode capture on RESETB deassertion
- configurable escalation of recurring ‘functional’ resets to ‘destructive’ reset

54.1.3 Reset Sources

The different reset sources are organized into two families: 'destructive' and 'functional'.

- A 'destructive' reset source is associated with an event related to a critical - usually hardware - error or dysfunction. When a 'destructive' reset event occurs, the full reset sequence is applied to the chip starting from PHASE0. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be unknown. 'Destructive' resets are
    - power-on reset
    - software 'destructive' reset
    - FCCU failure to react reset
    - 'functional' reset escalation
    - temperature sensor 'destructive' reset
    - voltage out of range 'destructive' reset

NOTE

POR occurs when the voltage starts from 0 or the voltage drops below the absolute minimum operating level. It remains active until the voltage is above the operating threshold. After that, the device is no longer in the POR state (although the POR flag remains set until software clears it), and other reset events can potentially occur.

- A 'functional' reset source is associated with an event related to a less-critical - usually non-hardware - error or dysfunction. When a 'functional' reset event occurs, a partial reset sequence is applied to the chip starting from PHASE1. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules (e.g., debug modules, flash modules) as well as the system memory content is preserved. 'Functional' resets are

- external reset
- start-up self test completed
- software 'functional' reset
- FCCU hard reaction
- FCCU soft reaction
- JTAG 'functional' reset
- temperature sensor 'functional' reset
- voltage out of range 'functional' reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e., PHASEn states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the IDLE phase. During this entire process, the MC_ME state machine is held in RESET mode. Only at the end of the reset sequence, when the IDLE phase is reached, does the MC_ME enter the DRUN mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt issued to the core (see 'Functional' Event Reset Disable Register (MC_RGM_FERD) and 'Functional' Event Alternate Request Register (MC_RGM_FEAR)).

54.2 External Signal Description

The MC_RGM interfaces to the bidirectional reset pin RESETB.

54.3 Memory Map and Register Definition

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the RGM_DES[8:15] register bits may be accessed as a word at address offset 0x000, as a half-word at address offset 0x002, or as a byte at address offset 0x003.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

MC_RGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	'Destructive' Event Status Register (MC_RGM_DES)	32	w1c	0000_0001h	54.3.1/2181
10	‘Destructive’ Event Reset Disable Register (MC_RGM_DERD)	32	R/W	0000_0000h	54.3.2/2183
30	Destructive’ Bidirectional Reset Enable Register (MC_RGM_DBRE)	32	R	0000_0000h	54.3.3/2185
300	'Functional' Event Status Register (MC_RGM_FES)	32	w1c	0000_0000h	54.3.4/2187
310	'Functional' Event Reset Disable Register (MC_RGM_FERD)	32	R/W	0000_0000h	54.3.5/2189
320	'Functional' Event Alternate Request Register (MC_RGM_FEAR)	32	R/W	0000_0000h	54.3.6/2191
330	'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE)	32	R/W	0180_0468h	54.3.7/2192
340	'Functional' Event Short Sequence Register (MC_RGM_FESS)	32	R/W	0000_0040h	54.3.8/2194
604	'Functional' Reset Escalation Threshold Register (MC_RGM_FRET)	8	R/W	0Fh	54.3.9/2196
608	'Destructive' Reset Escalation Threshold Register (MC_RGM_DRET)	8	R/W	00h	54.3.10/ 2197

54.3.1 'Destructive' Event Status Register (MC_RGM_DES)

This register contains the status of the 'destructive' reset sources. It can be accessed as read/write in either supervisor mode or test mode. It can be accessed as read-only in user mode. Register bits are cleared on write '1'.

This register is reset only on power-on.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							F_VOR_DEST	F_TSR_DEST	0						
W								w1c	w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						F_FIF	F_EDR	0		F_SUF	F_FFRR	F_SOFT_DEST	0		F_POR
W							w1c	w1c			w1c	w1c	w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MC_RGM_DES field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 F_VOR_DEST	Flag for voltage out of range 'destructive' reset 0 No voltage out of range 'destructive' reset event has occurred since the last clear 1 A voltage out of range 'destructive' reset event has occurred
8 F_TSR_DEST	Flag for temperature sensor 'destructive' reset 0 No temperature sensor 'destructive' reset event has occurred since the last clear 1 A temperature sensor 'destructive' reset event has occurred
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 F_FIF	Flag for flash initialization failure 0 No flash memory initialization failure event has occurred since the last clear 1 A flash memory initialization failure event has occurred
23 F_EDR	Flag for 'functional' reset escalation

Table continues on the next page...

MC_RGM_DES field descriptions (continued)

Field	Description
	0 No 'functional' reset escalation event has occurred since the last clear 1 A 'functional' reset escalation event has occurred
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 F_SUF	Flag for STCU unrecoverable fault 0 No STCU unrecoverable fault event has occurred since the last clear 1 An STCU unrecoverable fault event has occurred
27 F_FFRR	Flag for FCCU failure to react reset 0 No FCCU failure to react reset event has occurred since the last clear 1 A FCCU failure to react reset event has occurred
28 F_SOFT_DEST	Flag for software 'destructive' reset 0 No software 'destructive' reset event has occurred since the last clear 1 A software 'destructive' reset event has occurred
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 F_POR	Flag for Power-On reset 0 No power-on event has occurred since the last clear 1 A power-on event has occurred

54.3.2 ‘Destructive’ Event Reset Disable Register (MC_RGM_DERD)

This register provides dedicated bits to disable particular ‘destructive’ reset sources. It can be accessed as read/write in either supervisor mode or test mode. It can be accessed as read-only in user mode. Each byte can be written only once after a ‘destructive’ or power-on reset.

This register is reset only on power-on.

WARNING

It is important to clear the RGM_DES register before setting any of the bits in the RGM_DERD register to ‘1’. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							D_VOR_DEST	D_TSR_DEST	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						F_FIF	D_EDR	0		F_SUF	D_FFRR	D_SOF_DEST	0		D_POR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_RGM_DERD field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 D_VOR_DEST	Disable voltage out of range ‘destructive’ reset 0 A voltage out of range ‘destructive’ reset event triggers a reset sequence

Table continues on the next page...

MC_RGM_DERD field descriptions (continued)

Field	Description
8 D_TSR_DEST	Disable temperature sensor 'destructive' reset 0 A temperature sensor 'destructive' reset event triggers a reset sequence
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 F_FIF	Flash initialization failure 0 A flash memory initialization failure event triggers a reset sequence
23 D_EDR	Disable ‘functional’ reset escalation 0 A ‘functional’ reset escalation event triggers a reset sequence
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 F_SUF	STCU unrecoverable fault 0 An STCU unrecoverable fault event triggers a reset sequence
27 D_FFRR	Disable FCCU failure to react reset 0 A FCCU failure to react reset event triggers a reset sequence
28 D_SOF_DEST	Disable software ‘destructive’ reset 0 A software ‘destructive’ reset triggers a reset sequence
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 D_POR	Disable Power-On reset 0 A power-on event triggers a reset sequence

54.3.3 Destructive’ Bidirectional Reset Enable Register (MC_RGM_DBRE)

This register enables the generation of an external reset on ‘destructive’ reset. It can be accessed in read only in all modes. Since all ‘destrutcive’ resets assert RESETB, all the bits in this register are always ‘0’.

This register is reset only on power-on.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							BE_VOR_DEST	BE_TSR_DEST	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						F_FIF	BE_EDR	0		F_SUF	BE_FFRR	BE_SOF_DEST	0		BE_POR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_RGM_DBRE field descriptions

Field	Description
0–6 Reserved	This field is reserved.
7 BE_VOR_DEST	Bidirectional Reset Enable for voltage out of range ‘destructive’ reset 0 RESETB is asserted on a voltage out of range ‘destructive’ reset event

Table continues on the next page...

MC_RGM_DBRE field descriptions (continued)

Field	Description
8 BE_TSR_DEST	Bidirectional Reset Enable for temperature sensor 'destructive' reset 0 RESETB is asserted on a temperature sensor 'destructive' reset event if the reset is enabled
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 F_FIF	Flash initialization failure 0 RESETB is asserted on a flash memory initialization failure event
23 BE_EDR	Bidirectional Reset Enable for ‘functional’ reset escalation 0 RESETB is asserted on a ‘functional’ reset escalation event
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 F_SUF	STCU unrecoverable fault 0 RESETB is asserted on an STCU unrecoverable fault event
27 BE_FFRR	Bidirectional Reset Enable for FCCU failure to react reset 0 RESETB is asserted on a FCCU failure to react reset event
28 BE_SOF_DEST	Bidirectional Reset Enable for software ‘destructive’ reset 0 RESETB is asserted on asoftware ‘destructive’ reset event
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 BE_POR	Bidirectional Reset Enable for Power-On reset 0 RESETB is asserted on a Power-On reset event

54.3.4 'Functional' Event Status Register (MC_RGM_FES)

This register contains the status of the 'functional' reset sources, including those configured to not generate a reset sequence. It can be accessed as read/write in either supervisor mode or test mode. It can be accessed as read-only in user mode. Register bits are cleared on write '1' if the triggering event has already been cleared at the source.

This register is reset only on power-on.

NOTE

If a 'functional' reset source is configured to generate a SAFE mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles before it clears the associated RGM_FES status bit in order to avoid multiple SAFE mode requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the RGM_FES has been properly cleared, and if not, clear it again.

Address: 0h base + 300h offset = 300h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							F_VOR_FUNC	F_TSR_FUNC	0						
W								w1c	w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					F_JTAG_FUNC	0			F_FCCU_SOFT	F_FCCU_HARD	0	F_SOFT_FUNC	F_ST_DONE	0	F_EXR
W						w1c				w1c	w1c		w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_RGM_FES field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 F_VOR_FUNC	Flag for voltage out of range 'functional' reset 0 No voltage out of range 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A voltage out of range 'functional' reset event has occurred
8 F_TSR_FUNC	Flag for temperature sensor 'functional' reset 0 No temperature sensor 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A temperature sensor 'functional' reset event has occurred
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_FES field descriptions (continued)

Field	Description
21 F_JTAG_FUNC	Flag for JTAG 'functional' reset The EXTEST, HIGHZ, and CLAMP instructions cause a JTAG 'functional' reset event to occur, which sets this field to 1. 0 No JTAG 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A JTAG 'functional' reset event has occurred
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 F_FCCU_SOFT	Flag for FCCU soft reaction 0 No FCCU soft reaction event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU soft reaction event has occurred
26 F_FCCU_HARD	Flag for FCCU hard reaction reset 0 No FCCU hard reaction reset event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU hard reaction reset event has occurred
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 F_SOFT_FUNC	Flag for software 'functional' reset 0 No software 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A software 'functional' reset event has occurred
29 F_ST_DONE	Flag for self test completed 0 No self test completed event has occurred since either the last clear or the last power-on reset assertion 1 A self test completed event has occurred
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 F_EXR	Flag for external reset 0 No external reset event has occurred since either the last clear or the last power-on reset assertion 1 An external reset event has occurred

54.3.5 'Functional' Event Reset Disable Register (MC_RGM_FERD)

This register provides dedicated bits to disable 'functional' reset sources.When a 'functional' reset source is disabled, the associated 'functional' event will trigger either a SAFE mode request or an interrupt request (see 'Functional' Event Alternate Request Register (MC_RGM_FEAR)). It can be accessed as read/write in either supervisor mode or test mode. It can be accessed as read-only in user mode. Each byte can be written only once after power-on reset.

This register is reset only on power-on and any 'destructive' reset.

Warning

It is important to clear the RGM_FES register before setting any of the bits in the RGM_FERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0h base + 310h offset = 310h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							D_VOR_FUNC	D_TSR_FUNC	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					D_JTAG_FUNC	0			D_FCCU_SOFT	D_FCCU_HARD	0	D_SOFT_FUNC	D_ST_DONE	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_RGM_FERD field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 D_VOR_FUNC	Disable voltage out of range 'functional' reset

Table continues on the next page...

MC_RGM_FERD field descriptions (continued)

Field	Description
	0 A voltage out of range 'functional' reset event triggers a reset sequence 1 A voltage out of range 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC
8 D_TSR_FUNC	Disable temperature sensor 'functional' reset 0 A temperature sensor 'functional' reset event triggers a reset sequence 1 A temperature sensor 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_JTAG_FUNC
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 D_JTAG_FUNC	Disable JTAG 'functional' reset 0 A JTAG 'functional' reset event triggers a reset sequence 1 A JTAG 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_JTAG_FUNC
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 D_FCCU_SOFT	Disable FCCU soft reaction 0 A FCCU soft reaction event triggers a reset sequence
26 D_FCCU_HARD	Disable FCCU hard reaction reset 0 A FCCU hard reaction reset event triggers a reset sequence
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 D_SOFT_FUNC	Disable software 'functional' reset 0 A software 'functional' reset event triggers a reset sequence
29 D_ST_DONE	Disable self test completed 0 A self test completed event triggers a reset sequence
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.3.6 'Functional' Event Alternate Request Register (MC_RGM_FEAR)

This register defines an alternate request to be generated when a reset on a 'functional' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed as read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 320h offset = 320h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							AR_VOR_FUNC	AR_TSR_FUNC	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					AR_JTAG_FUNC	0									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_RGM_FEAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 AR_VOR_FUNC	Alternate Request for voltage out of range 'functional' reset 0 Generate a SAFE mode request on a voltage out of range 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a voltage out of range 'functional' reset event if the reset is disabled
8 AR_TSR_FUNC	Alternate Request for temperature sensor 'functional' reset 0 Generate a SAFE mode request on a temperature sensor 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor 'functional' reset event if the reset is disabled
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 AR_JTAG_FUNC	Alternate Request for JTAG 'functional' reset 0 Generate a SAFE mode request on a JTAG 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a JTAG 'functional' reset event if the reset is disabled
22–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.3.7 'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE)

This register enables the generation of an external reset on 'functional' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 330h offset = 330h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							BE_VOR_FUNC	BE_TSR_FUNC	0						
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					BE_JTAG_FUNC	0			BE_FCCU_SOFT	BE_FCCU_HARD	0	BE_SOFT_FUNC	BE_ST_DONE	0	BE_EXR
W																
Reset	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0	0

MC_RGM_FBRE field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 BE_VOR_FUNC	Bidirectional Reset Enable for voltage out of range 'functional' reset 0 RESET_B is asserted on a voltage out of range 'functional' reset event if the reset is enabled 1 RESET_B is not asserted on a voltage out of range 'functional' reset event
8 BE_TSR_FUNC	Bidirectional Reset Enable for temperature sensor 'functional' reset 0 RESET_B is asserted on a temperature sensor 'functional' reset event if the reset is enabled 1 RESET_B is not asserted on a temperature sensor 'functional' reset event
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_FBRE field descriptions (continued)

Field	Description
21 BE_JTAG_FUNC	Bidirectional Reset Enable for JTAG 'functional' reset 0 RESET_B is asserted on a JTAG 'functional' reset event if the reset is enabled 1 RESET_B is not asserted on a JTAG 'functional' reset event
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 BE_FCCU_SOFT	Bidirectional Reset Enable for FCCU soft reaction 0 RESET_B is asserted on a FCCU soft reaction event 1 RESET_B is not asserted on a FCCU soft reaction event
26 BE_FCCU_ HARD	Bidirectional Reset Enable for a FCCU hard reaction 0 RESET_B is asserted on a FCCU hard reaction reset event 1 RESET_B is not asserted on a FCCU hard reaction reset event
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 BE_SOFT_FUNC	Bidirectional Reset Enable for software 'functional' reset 0 RESET_B is asserted on a software 'functional' reset event if the reset is enabled 1 RESET_B is not asserted on a software 'functional' reset event
29 BE_ST_DONE	Bidirectional Reset Enable for self test completed 0 RESET_B is asserted on a self test completed event
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 BE_EXR	Bidirectional Reset Enable for external reset 0 RESET_B is asserted on an external reset event if the reset is enabled

54.3.8 'Functional' Event Short Sequence Register (MC_RGM_FESS)

This register defines which reset sequence will be done when a 'functional' reset sequence is triggered. The 'functional' reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2.

NOTE

This could be useful for fast reset sequence, for example to skip flash reset.

It can be accessed as read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 340h offset = 340h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							SS_VOR_FUNC	SS_TSR_FUNC	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SS_JTAG_FUNC	0			SS_FCCU_SOFT	SS_FCCU_HARD	0	SS_SOFT_FUNC	SS_ST_DONE	0	SS_EXR
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

MC_RGM_FESS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SS_VOR_FUNC	Short Sequence for voltage out of range 'functional' reset 0 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
8 SS_TSR_FUNC	Short Sequence for temperature sensor 'functional' reset

Table continues on the next page...

MC_RGM_FESS field descriptions (continued)

Field	Description
	0 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SS_JTAG_FUNC	Short Sequence for JTAG 'functional' reset 0 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SS_FCCU_SOFT	Short Sequence for FCCU soft reaction 1 The reset sequence triggered by a FCCU soft reaction event will start from PHASE3, skipping PHASE1 and PHASE2
26 SS_FCCU_ HARD	Short Sequence for a FCCU hard reaction 0 The reset sequence triggered by a FCCU hard reaction reset event will start from PHASE1
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 SS_SOFT_FUNC	Short Sequence for software 'functional' reset 0 The reset sequence triggered by a software 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a software 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
29 SS_ST_DONE	Short Sequence for self test completed 0 The reset sequence triggered by a self test completed event will start from PHASE1
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SS_EXR	Short Sequence for EXR External Reset 0 The reset sequence triggered by an EXR external reset event will start from PHASE1 1 The reset sequence triggered by an EXR external reset event if the reset is enabled will start from PHASE3, skipping PHASE1 and PHASE2

54.3.9 'Functional' Reset Escalation Threshold Register (MC_RGM_FRET)

This register sets the threshold for 'functional' reset escalation to a 'destructive' reset. It can be accessed as read/write in either supervisor mode or test mode. It can be accessed as read-only in user mode.

Writing a non-zero value to the FRET field will enable the 'functional' reset escalation function. Writing any value to this register will reset the 'functional' reset counter. See 'Functional' Reset Escalation for details on the 'functional' reset escalation function.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 604h offset = 604h

Bit	0	1	2	3	4	5	6	7
Read	0				FRET			
Write								
Reset	0	0	0	0	1	1	1	1

MC_RGM_FRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 FRET	'Functional' Reset Escalation Threshold - If the value of this field is 0, the 'functional' reset escalation function is disabled. Any other value is the number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register isn't written to beforehand.

54.3.10 'Destructive' Reset Escalation Threshold Register (MC_RGM_DRET)

This register sets the threshold for 'destructive' reset escalation to keep the chip in the reset state until the next power-on reset triggers a new reset sequence. The register can be accessed as read/write in either supervisor mode or test mode. It can be accessed as read-only in user mode.

Writing a non-zero value to the DRET field enables the 'destructive' reset escalation function. Writing any value to this register resets the 'destructive' reset counter. See 'Destructive' Reset Escalation for details on the 'destructive' reset escalation function.

This register is reset only on power-on.

Address: 0h base + 608h offset = 608h

Bit	0	1	2	3	4	5	6	7
Read	0				DRET			
Write								
Reset	0	0	0	0	0	0	0	0

MC_RGM_DRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_DRET field descriptions (continued)

Field	Description
4–7 DRET	'Destructive' Reset Escalation Threshold - If the value of this field is 0, the 'destructive' reset escalation function is disabled. Any other value is the number of 'destructive' resets that will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written beforehand.

54.4 Functional Description

54.4.1 Reset State Machine

The main role of the MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in the following table.

Table 54-1. MC_RGM Reset Implications

Source	What Gets Reset	External Reset Assertion[1]	Boot Mode Capture
power-on reset	all	yes	yes
'destructive' resets	all except some clock/reset management	yes	yes
external reset	all except some clock/reset management and debug	programmable	yes
'functional' resets	all except some clock/reset management, STCU, FCCU, and debug	programmable[2]	programmable
shortened external or shortened 'functional' resets	flip-flops except some clock/reset management, STCU, FCCU, and debug	programmable[2]	programmable[3]

1. 'external reset assertion' means that the RESETB pin is asserted by the MC_RGM from the start of the reset sequence until the end of reset PHASE3
2. the assertion of the external reset is controlled via the RGM_FBRE register
3. the boot mode is captured if the external reset is asserted

Note

> JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in the following figure.

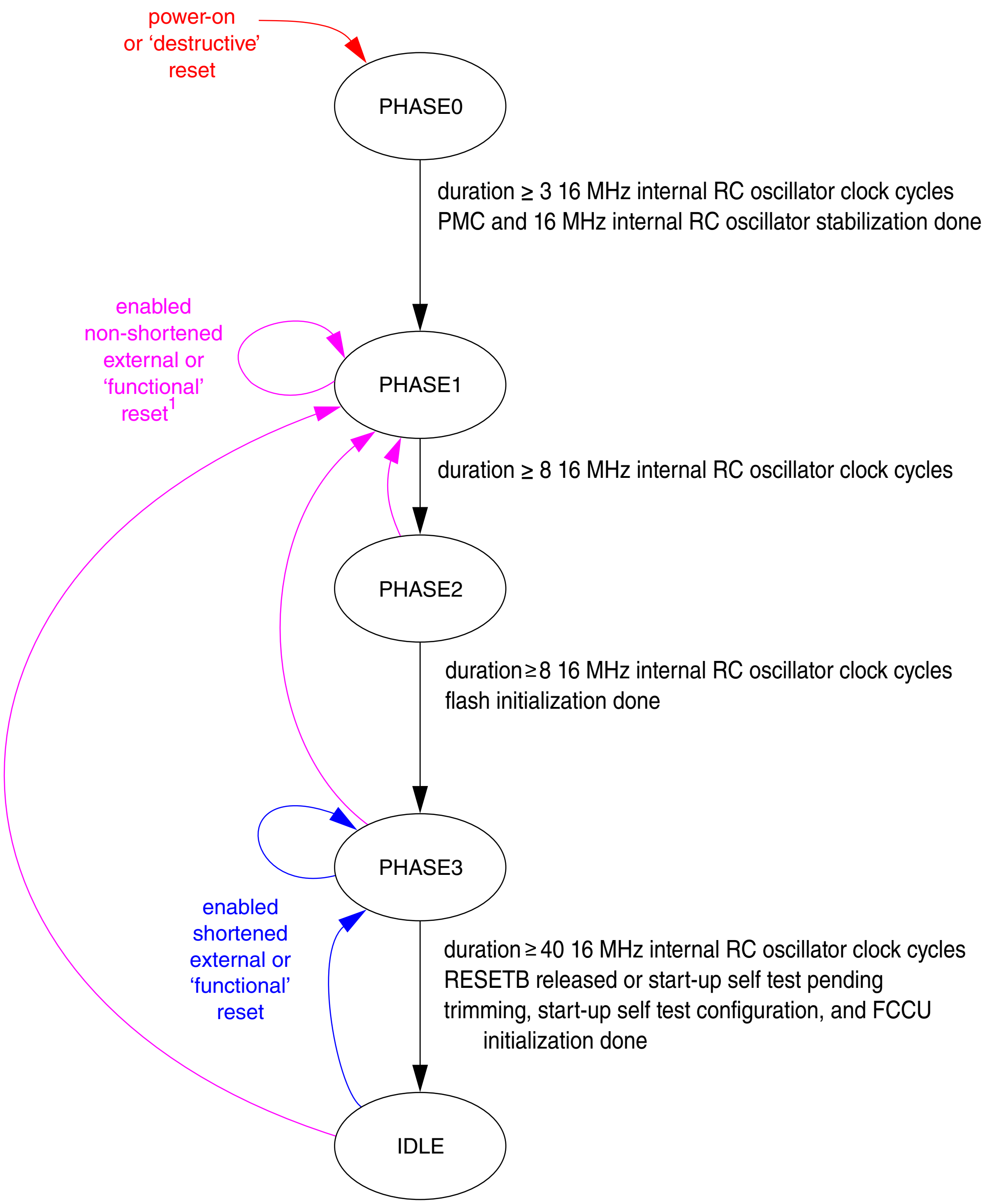

power-on
or 'destructive'
reset
PHASE0
duration ≥ 3 16 MHz internal RC oscillator clock cycles
PMC and 16 MHz internal RC oscillator stabilization done
enabled
non-shortened
external or
'functional'
reset[1]
PHASE1
duration ≥ 8 16 MHz internal RC oscillator clock cycles
PHASE2
duration≥8 16 MHz internal RC oscillator clock cycles
flash initialization done
PHASE3
enabled
shortened
external or
'functional'
reset
duration≥40 16 MHz internal RC oscillator clock cycles
RESETB released or start-up self test pending
trimming, start-up self test configuration, and FCCU
initialization done
IDLE

Figure 54-2. MC_RGM State Machine

54.4.1.1 PHASE0 Phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits PHASE0 and enters PHASE1 on verification of the following:

- all enabled 'destructive' resets have been processed

- all processes that need to be done in PHASE0 are completed
    - PMC and 16 MHz internal RC oscillator stabilization
- a minimum of 3 16 MHz internal RC oscillator clock cycles have elapsed since power-up completion and the last enabled 'destructive' reset event

54.4.1.2 PHASE1 Phase

This phase is entered either on exit from PHASE0 or immediately from PHASE2, PHASE3, or IDLE on a non-masked external or 'functional' reset event if it has not been configured to trigger a 'short' sequence. The reset state machine exits PHASE1 and enters PHASE2 on verification of the following:

- all enabled, non-shortened 'functional' resets have been processed
- a minimum of 8 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened 'functional' reset event

54.4.1.3 PHASE2 Phase

This phase is entered on exit from PHASE1. The reset state machine exits PHASE2 and enters PHASE3 on verification of the following:

- all processes that need to be done in PHASE2 are completed
    - flash initialization
- a minimum of 8 16 MHz internal RC oscillator clock cycles have elapsed since entering PHASE2

54.4.1.4 PHASE3 Phase

This phase is a entered either on exit from PHASE2 or immediately from IDLE on an enabled, shortened 'functional' reset event. The reset state machine exits PHASE3 and enters IDLE on verification of the following:

- all processes that need to be done in PHASE3 are completed
    - trimming, start-up self test configuration, and FCCU initialization

- a minimum of 40 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled, shortened 'functional' reset event
- RESETB is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence

54.4.1.5 IDLE Phase

This is the final phase and is entered on exit from PHASE3. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

54.4.2 'Destructive' Resets

A 'destructive' reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed.

The status flag associated with a given 'destructive' reset event (RGM_DES.F_<destructive reset> bit) is set when the 'destructive' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A 'destructive' reset will trigger a reset sequence starting from the beginning of PHASE0.

54.4.3 External Reset

The MC_RGM manages the external reset coming from RESETB. The detection of a falling edge on RESETB will start the reset sequence.

The status flag associated with the external reset falling edge event (RGM_FES.F_EXR bit) is set when the external reset is asserted and the power-on reset is not asserted.

An external reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by the external reset. When RGM_FESS.SS_EXR is set, the external reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially when an external reset should not reset the flash.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event
- an external reset event if configured via the RGM_FBRE register to assert the external reset
- a 'functional' reset event configured via the RGM_FBRE register to assert the external reset

In this case, RESETB is asserted until all conditions for exiting PHASE3 have been met with the exception of the RESETB assertion check.

In addition, the MC_RGM asserts RESETB while the start-up self test is being executed.

The RESETB input is disabled immediately as of the RESETB output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4µs after the RESETB output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

54.4.4 'Functional' Resets

A 'functional' reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given 'functional' reset event (RGM_FES.F_<functional reset> bit) is set when the 'functional' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'functional' reset can be optionally disabled by software writing bit RGM_FERD.D_<functional reset>.

Note

> The RGM_FERD register can be written only once between two power-on reset events.

An enabled 'functional' reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by a 'functional' reset. When RGM_FESS.SS_<functional reset> is set, the associated 'functional' reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially in case a 'functional' reset should not reset the flash module.

See the MC_ME chapter for details on the STANDBY0 and DRUN modes.

54.4.5 Alternate Event Generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the RGM_FERD and RGM_FEAR registers as shown in the following table.

Table 54-2. MC_RGM Alternate Event Selection

RGM_FERD Bit Value	RGM_FEAR Bit Value	Generated Event
0	X	reset
1	0	SAFE mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (that is, at the reset source that caused the alternate request) and also clearing the appropriate RGM_FES status bit.

Note

SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock; therefore, if the system clock was disabled at the time of the event, the interrupt request is not asserted until the system clock is re-enabled.

Note

If a masked 'destructive' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE0, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

Note

If a masked 'functional' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE1, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

54.4.6 'Functional' Reset Escalation

'Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' or external resets has occurred between software writes to the RGM_FRET register. This function is enabled by writing a non-zero value to the FRET field of this register.

Once 'functional' reset escalation has been enabled, the MC_RGM increases a counter on each 'functional' or external reset which causes a reset sequence to be initiated (i.e., entrance into PHASE1 or PHASE3 from the IDLE phase). This counter is cleared on a write of any value to the RGM_FRET register and on any power-on or 'destructive' reset. If the counter reaches the value in the RGM_FRET register's FRET field, the MC_RGM starts a reset sequence at PHASE0 and asserts the F_EDR bit in the RGM_DES register.

54.4.7 'Destructive' Reset Escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until the power-on triggers a reset sequence if a number of 'destructive' resets has occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

After 'destructive' reset escalation has been enabled, the MC_RGM increases a counter on each 'destructive' reset that is enabled to increment the escalator and that causes a reset sequence to be initiated (i.e., entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (i.e., entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the value in the RGM_DRET register's DRET field, the MC_RGM enters reset PHASE0 and remains there until the next power-on reset occurs.

On a power-on reset, the dest_rst_escalation output is deasserted. It is asserted by the MC_RGM when the counter reaches the value in the RGM_DRET register’s DRET field. It is not deasserted until the next power-on reset.

54.4.8 Boot Mode Capturing

The MC_RGM samples FAB/ABS whenever RESETB is asserted until five 16 MHz internal RC oscillator clock cycles before its deassertion edge. The result of the sampling is used at the beginning of reset PHASE3 for boot mode selection and is retained after RESETB has been deasserted for subsequent boots after reset sequences during which RESETB is not asserted.

NOTE

In order to ensure that the boot mode is correctly captured, the application needs to apply the valid boot mode value the entire time that RESETB is asserted and to keep applying it for 5 µs after RESETB is deasserted. For non-alternate boot modes, it is recommended to use a dedicated pin for the FAB pin and to keep it asserted to ‘0’ at all times.

NOTE

RESETB can be asserted as a consequence of the internal reset generation. This will force re-sampling of the boot mode pins. (See Table 54-1 for details.)

Chapter 55
Boot Assist Module (BAM)

55.1 Overview

The Boot Assist Module (BAM) is a block of read-only memory containing VLE code which is executed according to the boot mode of the device. The code stored in the BAM is not executed when booting in Single Chip mode, except when entering the "Static mode" in case no valid bootable section has been found.

The BAM downloads code into internal SRAM through the following serial protocols and executes it afterwards:

- UART (using LINFlexD)
- CAN (using FlexCAN)

Depending on the selected boot mode (see Entering boot modes), any download is performed with a fixed baud rate.

55.2 Features

The BAM provides the following features:

- Programmable 64-bit password protection for serial boot mode
- Serial boot loads the application boot code from a FlexCAN or LINFlexD bus into internal SRAM
- Censorship protection for internal flash module

55.3 Memory map

The BAM code resides in 8 KB of ROM mapped from address 0xFFFF_C000. The address space and memory used by the BAM is shown in the following table.

Table 55-1. BAM memory organization

Entity	Address
BAM entry point	0xFFFF_C000
RAM area used by the BAM code (do not use)	0x4000_0000–0x4000_00FF
Downloaded code base address	0x4000_0100

The RAM location where to download the code can be any 8 byte-aligned location in the SRAM starting from the address 0x4000_0100.

Caution

Do not use the RAM area used by the BAM code as indicated in the preceding table.

55.4 Functional description

55.4.1 Entering boot modes

The device detects the boot mode based on external pins and device status. The following sequence applies (see the following figure):

1. To boot either from FlexCAN or LINFlexD, the device must be forced into an Alternate Boot Loader Mode via the FAB (Force Alternate Boot Mode) pin which must be asserted before initiating the reset sequence. The type of alternate boot mode is selected according to the ABS (Alternate Boot Selector) pins (see the following table).
2. If FAB is not asserted, the device boots from the first flash-memory sector which contains a valid boot signature.
3. If no flash memory sector contains a valid boot signature, the device will go into static mode.

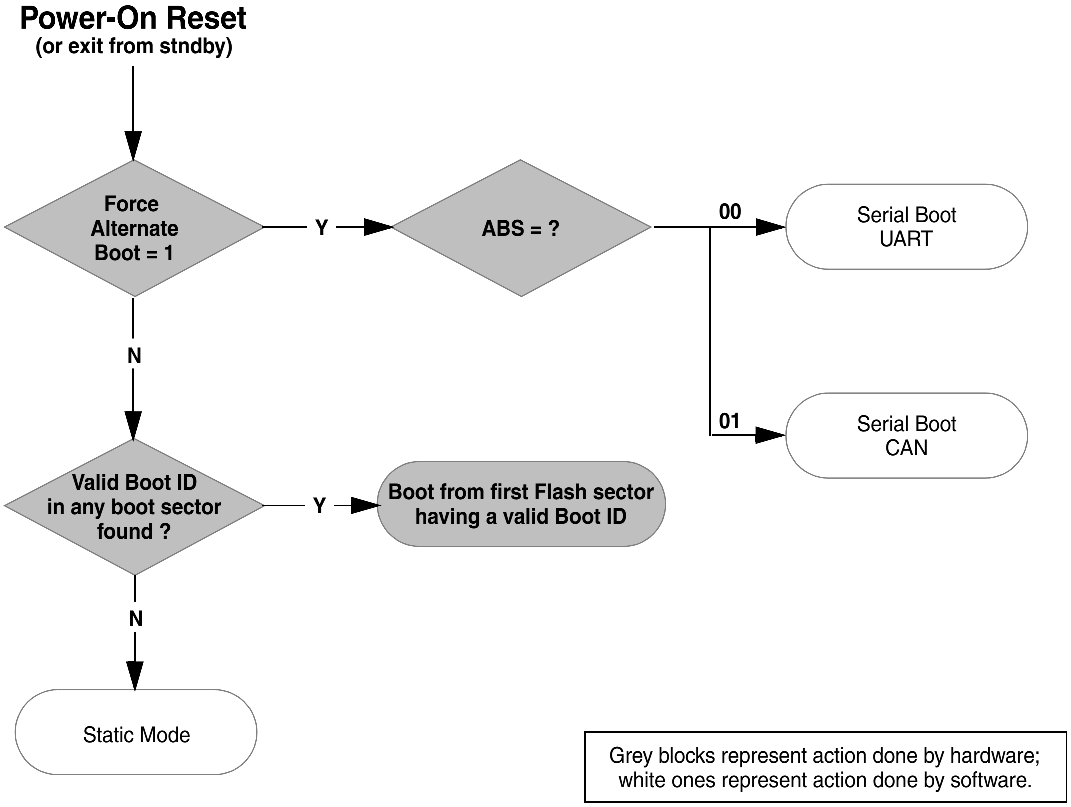

Power-On Reset
(or exit from stndby)
Force Alternate Boot = 1
Y
ABS = ?
00
Serial Boot UART
01
Serial Boot CAN
N
Valid Boot ID in any boot sector found ?
Y
Boot from first Flash sector having a valid Boot ID
N
Static Mode
Grey blocks represent action done by hardware; white ones represent action done by software.

Figure 55-1. Boot mode selection

Table 55-2. Hardware configuration to select boot mode

FAB	ABS[2,0]	Standby-RAM Boot Flag	Boot ID	Boot Mode
1	00	0	-	UART
1	01	0	-	CAN

55.4.2 Boot through BAM

55.4.2.1 Executing BAM

Single chip boot mode (selecting the first bootable flash memory sector) is managed by hardware and BAM does not participate in it.

BAM is executed only in one or both of the following cases:

- Serial boot mode has been selected by Force Alternate Boot (FAB) pin.
- Hardware has not found a valid Boot ID in any flash memory boot locations.

If one of these conditions is true, the device fetches code at location 0xFFFF_C000 and the BAM application starts.

55.4.2.2 BAM software flow

The following figure describes the BAM logic flow.

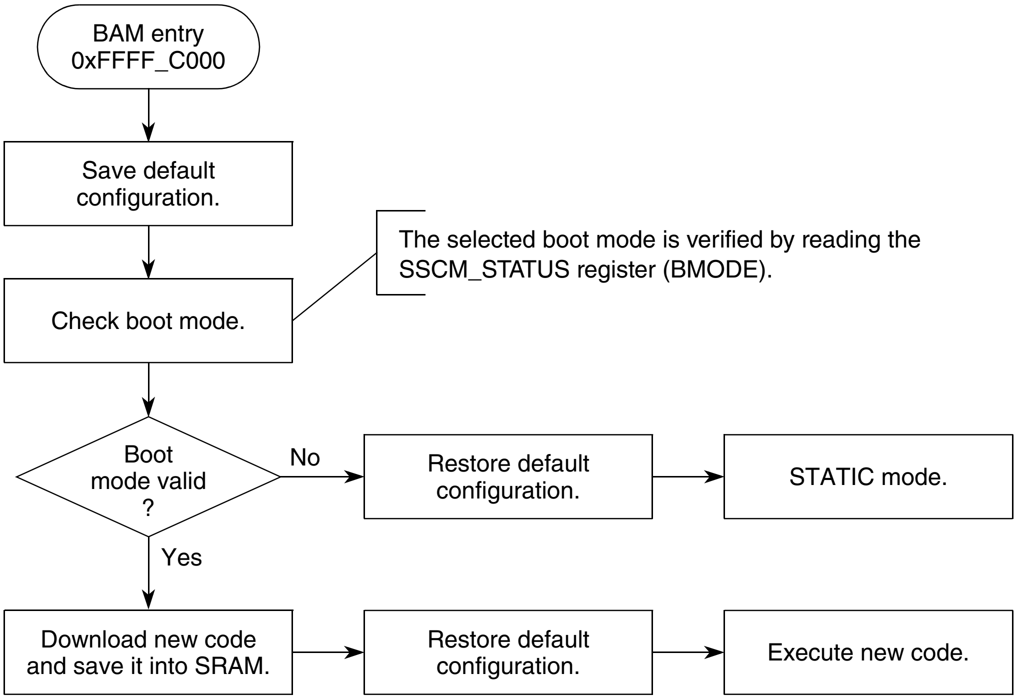

BAM entry
0xFFFF_C000
Save default configuration.
The selected boot mode is verified by reading the SSCM_STATUS register (BMODE).
Check boot mode.
Boot mode valid ?
No
Restore default configuration.
STATIC mode.
Yes
Download new code and save it into SRAM.
Restore default configuration.
Execute new code.

Figure 55-2. BAM logic flow

The first action is to save the initial device configuration. In this way it is possible to restore the initial configuration after downloading the new code but before executing it. This allows the new code to be executed as the device was just coming out of reset.

The SSCM_STATUS[BMODE] field indicates which boot has to be executed (see the following table).

If BMODE field shows either a single chip value (011) or the reserved values, the boot mode is not considered valid and the BAM pushes the device into static mode.

In all other cases data is downloaded in serial boot mode and saved into the correct SRAM location.

Table 55-3. Fields of SSCM_STATUS register read by BAM to detect the chosen boot mode

Field	Description
BMODE	BMODE Device Boot Mode 001 CAN Serial Boot Loader using FlexCAN

Table 55-3. Fields of SSCM_STATUS register read by BAM to detect the chosen boot mode

Field	Description
	010 UART Serial Boot Loader using LINFlexD 011 Single Chip other values are reserved

Then, the initial device configuration is restored and the code jumps to the address provided for the downloaded code. At this point BAM has just finished its task.

If there is any error (that is, communication error, wrong boot selected, etc.), BAM restores the default configuration and puts the device into static mode. Static mode means the device enters the low power mode SAFE and the processor executes a wait instruction. It is needed if the device can not boot in the mode which was selected. During BAM execution and after, the mode reported by the field S_CURRENT_MODE of the register ME_GS in the module MC_ME Module is "DRUN".

55.4.2.3 BAM resources

BAM uses/initializes the following MCU resources:

- MC_ME and MC_CGM modules to initialize mode and clock sources
- CAN_0, LINFlex_0 and their pins when performing serial boot mode
- SWT is disabled in case of a serial boot mode or when entering static mode
- PIT for some time measurement when performing serial boot mode
- SSCM to check the boot mode and during password check (see Table 55-3)
- External oscillator (XOSC)
- SIUL to perform programming of the required GPIO pins

The frequency of the external oscillator defines the baud rate for serial interfaces used to download the user application. For a LINFlexD transmission, the selected baud rate is f_{XOSC} / 832. For a FlexCAN transmission, the selected baud rate is f_{XOSC} / 40.

55.4.2.4 Download and execute the new code

From high level perspective, the download protocol follows these steps:

1. Transmit 64-bit password.
2. Transmit start address, size of downloaded code in bytes and VLE bit.
3. Transmit download data.
4. Execute code from start address.

Each step must be completed before the next step starts.

The communication is done in half duplex manner; any transmission from the host is followed by the MCU transmission.

- Host sends data to MCU and starts waiting
- MCU echoes to host the data received
- Host verifies if echoes are correct
    - if data is correct, the host can continue to send data
    - if data is not correct, the host stops to transmit and MCU must be reset

All multi-byte data structures are sent with MSB first.

A more detailed description of these steps follows.

NOTE

> There must be no stringent timeout condition while the host waits for the echoed data from the MCU; especially after step 2. After receiving the actual size of the data to be transmitted, the BAM must initialize the memory receiving the transmitted data with 64 bit writes to avoid ECC errors due to uninitialized memory.

55.4.2.5 Download 64-bit password and password check

The first 64 bits received represent the password. This password is sent to the Password Check procedure which verifies whether it is correct. The actual processing of the provided password by the Password Check procedure is described in the following table. The Password Check procedure performs the check with a time limiting condition that ensures no more than one check occurs within a certain timeframe. The corresponding timeout by the hardware performing the check works in parallel to the actual download to minimize the download time. In case of secured flash memory, after the timer that locks the hardware checking the password expires:

- If the correct password was supplied, flash memory is now unsecured and BAM continues its task.
- If an incorrect password was supplied, flash memory is still secured; BAM puts the device into static mode.

Table 55-4. Password Check procedure

Flash Secured	Public Password	Password requested	Comment
Yes	No	Password must match the password recorded in the corresponding DCF client	Flash is unsecured when providing the correct private password.
Yes	Yes	FEED_FACE_CAFE_BEEF	Flash is unsecured when providing the correct public password.
No	No	(password is not verified)	There is no benefit in checking the password, since the Flash is readable.
No	Yes	FEED_FACE_CAFE_BEEF	Password is checked, Flash is already unsecured

55.4.2.6 Download start address, VLE bit and code size

The next 8 bytes received by the MCU contain a 32-bit Start Address, the VLE mode bit and a 31-bit code Length as shown in the following figure.

The VLE bit (Variable Length Instruction) is used to indicate for which instruction set the code has been compiled. The BAM supports the download of VLE code.

The Start Address defines where the received data will be stored and where the MCU will branch after the download is finished. The two LSB bits of the Start Address are ignored by the BAM program, such that the loaded code should be 32-bit word aligned.

The Code Length defines the number of data bytes to be loaded.

<table>
<tr><td colspan="2">START_ADDRESS[31:16]</td></tr>
<tr><td colspan="2"></td></tr>
</table>

<table>
<tr><td colspan="2">START_ADDRESS[15:0]</td></tr>
<tr><td colspan="2"></td></tr>
</table>

<table>
<tr><td>VLE</td><td>CODE_LENGTH[30:16]</td></tr>
<tr><td colspan="2"></td></tr>
</table>

<table>
<tr><td colspan="2">CODE_LENGTH[15:0]</td></tr>
<tr><td colspan="2"></td></tr>
</table>

Figure 55-3. Start address, VLE bit and download size in bytes

55.4.2.7 Download data

Each byte of data received is stored into device's SRAM, starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until the number of bytes of data received matches the number of bytes specified in the previous protocol step.

Since the SRAM is protected by a 64-bit wide Error Correction Code (ECC), the BAM performs a series of 64 bit writes to initialize the SRAM are required for the downloaded data. The actual writes are performed in chunks of 32-bit writes. If the last byte received does not fall onto a 32-bit boundary, the BAM fills it with 0 bytes.

Since the SRAM is protected by 32-bit wide Error Correction Code (ECC), BAM always writes bytes into SRAM grouped into 32-bit words. If the last byte received does not fall onto a 32-bit boundary, the BAM fills it with 0 bytes.

Finally a "dummy" word (0x0000_0000) is written to avoid a possible ECC error during core prefetch.

55.4.2.8 Execute code

The BAM program waits for the last echo message transmission being completed.

Then it restores the initial MCU configuration and jumps to the code loaded at Start Address which was received in step 2 of the protocol.

At this point BAM has finished its tasks and MCU is controlled by new code executing from SRAM.

55.4.3 UART Boot

55.4.3.1 Configuration

Boot using the UART protocol is implemented by LINFlex_0 module. Pins used are:

- LINFlex_TX corresponds to pin B[2]
- LINFlex_RX corresponds to pin B[3].

The LINFlexD controller is configured to operate at a baud rate of f_{XOSC} / 832, using 8 bit data frame without parity bit and 1 stop bit.

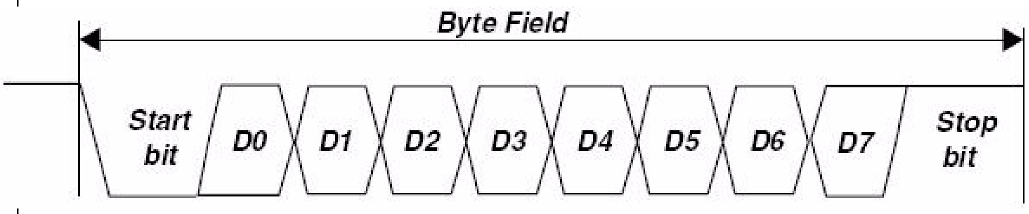


Figure 55-4. LINFlexD bit timing in UART mode

55.4.3.2 Protocol

The following table summarizes the protocol and BAM action during this boot mode.

Table 55-5. UART boot mode download protocol

Protocol step	Host sent message	BAM response message	Action
1	64-bit password (MSB first)	64-bit password	Password checked for validity and compared against stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + 31-bit number of bytes (MSB first)	VLE bit + 31-bit number of bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	8 bits of raw binary data	4 x 8 bits of data are packed into 32-bit words. These words are saved into SRAM starting from the "Load address". "Load address" increments until the number of data received and stored matches the size as specified in the previous step.
5	none	none	Branch to downloaded code

55.4.4 CAN Boot

55.4.4.1 Configuration

Boot using the CAN protocol is implemented by FlexCAN_0 module. Pins used are:

- CAN_TX corresponds to pin B[0]
- CAN_RX corresponds to pin B[1].

Boot from FlexCANuses the system clock driven by the external oscillator.

The FlexCAN controller is configured to operate at a baud rate = system clock frequency/40.

It uses the standard 11 bit identifier format detailed in FlexCAN 2.0A specification.

FlexCAN controller bit timing is programmed with 10 time quanta, and the sample point is 2 time quanta before the end, as shown in the following figure.

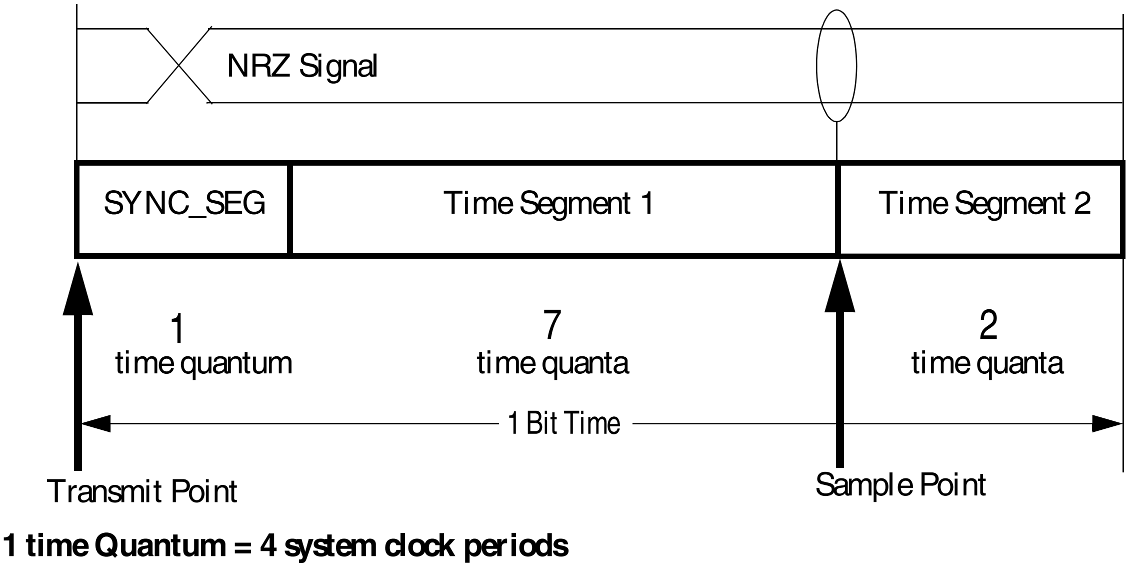


Figure 55-5. FlexCAN bit timing

55.4.4.2 Protocol

The following table summarizes the protocol and BAM action during this boot mode. All data is transmitted byte wise.

Table 55-6. FlexCAN boot mode download protocol

Protocol step	Host sent message	BAM response message	Action
1	FlexCAN ID 0x011+ 64-bit password	FlexCAN ID 0x001+ 64-bit password	Password checked for validity and compared against stored password.
2	FlexCAN ID 0x012+ 32-bit store address+ VLE bit+ 31-bit number of bytes	FlexCAN ID 0x002+ 32-bit store address+ VLE bit+ 31-bit number of bytes	Load address is stored for future use. Size of download are stored for future use. Verify VLE bit.

Table continues on the next page...

Table 55-6. FlexCAN boot mode download protocol (continued)

Protocol step	Host sent message	BAM response message	Action
3	FlexCAN ID 0x013+ 8 to 64 bits of raw binary data	FlexCAN ID 0x003+ 8 to 64 bits of raw binary data	4 x 8 bits of data are packed into 32-bit words. These words are saved into SRAM starting from the "Load address". "Load address" increments until the number of data received and stored matches the size as specified in the previous step.
4	none	none	Branch to downloaded code

55.4.5 Inhibiting BAM operation

Under certain circumstances, you may want to inhibit BAM operation. To do this, set to 1 the RAE field in the SSCM_ERROR register. After this change, any attempt to access the memory range occupied by the BAM results in an access error.

55.4.6 Interrupts

No interrupts are generated or enabled by the BAM.

Chapter 56
System Status and Configuration Module (SSCM)

56.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

56.1.1 Overview

The System Status and Configuration Module (SSCM) is pictured in the figure below.

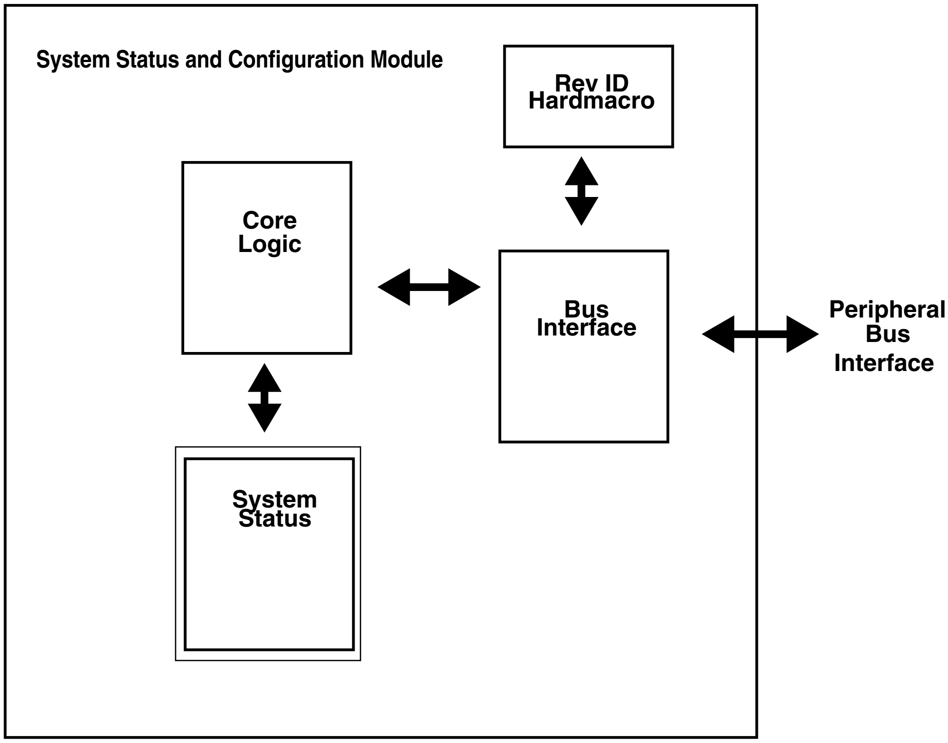


Figure 56-1. SSCM block diagram

56.1.2 Glossary

The table below shows a glossary of terms used through out the text.

Table 56-1. Glossary

Term	Definition
0011b	Binary numbers
0Fh	Hexadecimal numbers
Pin	External physical connection.
Signal	Electronic construct whose state or change in state conveys information.
X	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded X001b, the state of the first bit is a don't care.
y	y is used for a place holder of 1 hex digit for unknown values.

56.1.3 Features

The SSCM includes these distinct features:

- System Configuration and Status
  - Device Mode and System Status
  - Determine primary boot vector
- Debug Status Port enable and selection
- Bus abort enable/disable
- Peripheral abort enable/disable

56.1.4 Modes of operation

The SSCM operates identically in all system modes.

56.2 External signal description

The SSCM provides a set of pins for the debug port (see Debug Status Port Register (DEBUGPORT) for details).

56.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

The table below shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

NOTE

All registers are accessible via 8-bit, 16-bit or 32-bit reads and/or writes. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit read or write, SSCM Base + 02h, but performing a 16-bit access to SSCM Base + 0003h is illegal (that is, you should not make any transfer that is not aligned to burst size boundary).

NOTE

SSCM does not generate transfer errors in all memory holes.

SSCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SSCM System Status (SSCM_STATUS)	16	R	See section	56.3.1/2222
2	SSCM System Memory and ID Register (SSCM_MEMCONFIG)	16	R	See section	56.3.2/2224
6	SSCM Error Configuration Register (SSCM_ERROR)	16	R/W	0000h	56.3.3/2225
8	SSCM Debug Status Port Register (SSCM_DEBUGPORT)	16	R/W	0000h	56.3.4/2225
20	User Option Status Register (SSCM_UOPS)	32	R	See section	56.3.5/2227
28	Processor Start Address Register (SSCM_PSA)	32	R	See section	56.3.6/2227

56.3.1 SSCM System Status (SSCM_STATUS)

The System Status register reflects the current state of the system.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	0	CER	CERS	0	NXEN	PUB	SEC	0
Write		w1c	w1c					
Reset	0	0	0	0	*	*	*	0

Bit	8	9	10	11	12	13	14	15
Read	BMODE			VLE	0	0	0	
Write								
Reset	*	*	*	*	0	0	0	0

* Notes:

- VLE field: Reset value depends on the associated option bits in flash memory.
- BMODE field: Reset value depends on the device status after leaving reset.
- SEC field: Reset value depends on the device status after leaving reset.
- PUB field: Reset value depends on the device status after leaving reset.
- NXEN field: Reset value depends on the device status after leaving reset.

SSCM_STATUS field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error. If a non-permanent error is detected, this bit can be cleared by writing a 1. Refer to CERS-DCF Mechanism section. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct
2 CERS	Configuration Error for Safe DCF Clients This field indicates that the SSCM has detected a configuration error during reset while loading the Safe DCF Clients (DCF client with SPRD ADDR enabled, Triple Voting Enabled). If a non-permanent error is detected, this bit can be cleared by writing a 1. Refer to CERS-DCF Mechanism section. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SSCM_STATUS field descriptions (continued)

Field	Description
4 NXEN	Processor 0 Nexus enable status 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
5 PUB	Public Serial Access Status This bit indicates whether serial boot mode with public password is allowed. **NOTE:** Reset value depends on the device status after leaving reset. 0 Serial boot mode with private flash memory password is allowed. 1 Serial boot mode with public password is allowed.
6 SEC	Security Status This bit reflects the current security state of the Flash. **NOTE:** Reset value depends on the device status after leaving reset. 0 The flash memory is not secured 1 The flash memory is secured
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 BMODE	Device Boot Mode This field is only updated during reset. 001 CAN Serial Boot Loader 010 SCI Serial Boot Loader 011 Single Chip(no Boot flash memory)
11 VLE	Variable Length Instruction Mode When booting in SC mode, this field indicates that the code stored in the flash memory is using the VLE instruction set. The value of this field is determined by the RCHW field of the flash memory boot sector. **NOTE:** Reset value depends on the associated option bits in flash memory. 0 SC Boot Location contains standard PPC code 1 SC Boot Location contains VLE code
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.3.2 SSCM System Memory and ID Register (SSCM_MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG ID.

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read	JPIN							
Write								
Reset	*	*	*	*	*	*	*	*

Bit	8	9	10	11	12	13	14	15
Read	JPIN		IVLD	MREV				DVLD
Write								
Reset	*	*	*	*	*	*	*	*

* Notes:
- DVLD field: Reset value depends on boot mode and security status.
- MREV field: Reset value can be found in the SSCM Configuration section.
- IVLD field: Reset value depends on boot mode and security status.
- JPIN field: Reset value reflects the JTAG ID of the device.

SSCM_MEMCONFIG field descriptions

Field	Description
0–9 JPIN	JTAG Part ID Number **NOTE:** Reset value reflects the JTAG ID of the device.
10 IVLD	Instruction flash memory Valid This bit identifies whether or not the on-chip instruction flash memory is accessible in the system memory map. The Instruction flash memory may not be accessible due to security limitations, or because there is no instruction flash memory in the system. **NOTE:** Reset value depends on boot mode and security status. 0 Instruction Flash is not accessible 1 Instruction Flash is accessible
11–14 MREV	Minor Mask Revision
15 DVLD	Data flash memory Valid This bit identifies whether or not the on-chip data flash memory is visible in the system memory map. The data flash memory may not be accessible due to security limitations, or because there is no data flash memory in the system. **NOTE:** Reset value depends on boot mode and security status.

Table continues on the next page...

SSCM_MEMCONFIG field descriptions (continued)

Field	Description
	0 Data Flash is not visible 1 Data Flash is visible

56.3.3 SSCM Error Configuration Register (SSCM_ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0														PAE	RAE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSCM_ERROR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 PAE	Peripheral Bus Abort Enable. This bit enables bus aborts on: • any access to a peripheral slot that is not used on the device. • This feature is intended to aid in debugging when developing application code. • illegal accesses to off-platform peripherals. • On platform peripherals cannot be controlled by SSCM peripheral abort. 0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception
15 RAE	Register Bus Abort Enable. This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. This feature is intended to aid in debugging when developing application code. **NOTE:** Transfers to peripheral bus resources may be aborted even before they reach the peripheral bus (in example, at the AIPS level). In this case, the PAE and RAE register bits have no effect on the abort. 0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception

56.3.4 SSCM Debug Status Port Register (SSCM_DEBUGPORT)

The Debug Status Port register is used to (optionally) provide debug data on a set of pins.

Table 56-2. Debug status port modes

Pin [1]	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
0	STATUS[0]	STATUS[8]	MEMCONFIG[0]	MEMCONFIG[8]	Reserved	Reserved	Reserved
1	STATUS[1]	STATUS[9]	MEMCONFIG[1]	MEMCONFIG[9]	Reserved	Reserved	Reserved
2	STATUS[2]	STATUS[10]	MEMCONFIG[2]	MEMCONFIG[10]	Reserved	Reserved	Reserved
3	STATUS[3]	STATUS[11]	MEMCONFIG[3]	MEMCONFIG[11]	Reserved	Reserved	Reserved
4	STATUS[4]	STATUS[12]	MEMCONFIG[4]	MEMCONFIG[12]	Reserved	Reserved	Reserved
5	STATUS[5]	STATUS[13]	MEMCONFIG[5]	MEMCONFIG[13]	Reserved	Reserved	Reserved
6	STATUS[6]	STATUS[14]	MEMCONFIG[6]	MEMCONFIG[14]	Reserved	Reserved	Reserved
7	STATUS[7]	STATUS[15]	MEMCONFIG[7]	MEMCONFIG[15]	Reserved	Reserved	Reserved

[1] All signals are active high, unless otherwise noted

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7
Read	Reserved							Reserved
Write	▨							
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	Reserved					DEBUG_MODE		
Write	▨							
Reset	0	0	0	0	0	0	0	0

SSCM_DEBUGPORT field descriptions

Field	Description
0–6 Reserved	This field is reserved.
7 Reserved	This field is reserved.
8–12 Reserved	This field is reserved.
13–15 DEBUG_MODE	Debug Status Port Mode. This field selects the alternate debug functionality for the Debug Status Port Table 56-2 describes the functionality of the Debug Status Port in each mode. 000 Undefined 001 Mode 1 selected 010 Mode 2 selected

Table continues on the next page...

SSCM_DEBUGPORT field descriptions (continued)

Field	Description
	011 Mode 3 selected 100 Mode 4 selected 101 Mode 5 selected 110 Mode 6 selected 111 Mode 7 selected

56.3.5 User Option Status Register (SSCM_UOPS)

NOTE

Reset value depends on the state of the user option bits in flash memory at time of reset.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UOPT																															
W																																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- Reset value depends on the state of the user option bits in flash memory at time of reset.
- UOPT field: Reset value depends on the state of the user option bits in flash at time of reset.

SSCM_UOPS field descriptions

Field	Description
0–31 UOPT	Shows the values read from the User Option Bits location in the flash memory. The value is set by programming the UOPT DCF record; if no such record exists, all bits are 1.

56.3.6 Processor Start Address Register (SSCM_PSA)

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADR																															
W																																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- SADR field: Reset value is the start address

SSCM_PSA field descriptions

Field	Description
0–31 SADR	Processor Start Address The boot processor will start executing application code from this address.In SC mode this indicates the location determined by the RCHW search.In the case of a serial download, this register will point to the BAM start location.

56.4 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

56.4.1 DCF mechanism

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device will process these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

Each record is 64 bits in length. Table 56-3 shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records should be processed.

Table 56-3. DCF start record

0:31
05AA 55AFh
0:31
0000 0000h

The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - See Table 56-4. In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Table 56-4. Format of a DCF data record

0:31
WDATA[31:0]

0:14	15:29	30	31
CS[14:0]	ADDR[16:2]	PRTY	STOP

The stop record indicates that processing should stop. The general format of the stop record is shown in Table 56-5. Only the STOP bit needs to be 1 in order to form a stop record – All other bits are ignored. An unprogrammed location in UTEST flash will be interpreted as a stop record.

Table 56-5. DCF stop record

0:30	31
rsvd	1

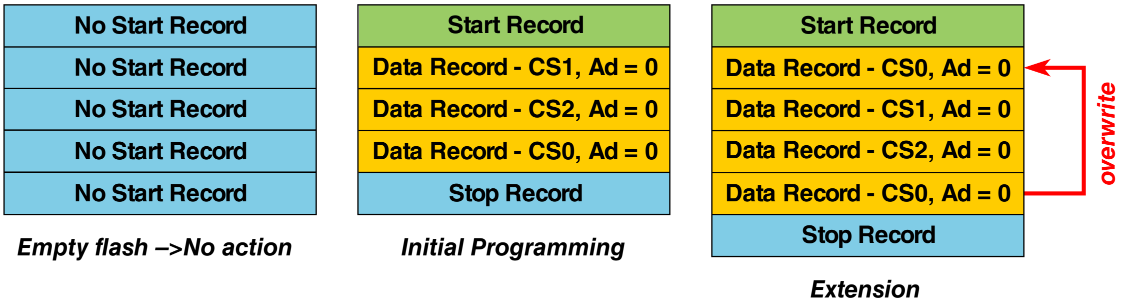
0:31
rsvd

If *n* data records are to be stored in UTEST, the data structure must be as shown in Table 56-6.

Table 56-6. Series of DCF records in UTEST

ADDR offset	DATA			
00h	05AA 55AFh			
04h	0000 0000h			STOP=0
08h	WDATA[31:0]			
0Ch	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
10h	WDATA[31:0]			
14h	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
…	…			
8n - 1 + 0h	reserved			
8n - 1 + 4h	reserved			1
8n + 0h				
8n + 4h				

There must never be an unprogrammed record in the data structure, as that would be interpreted as a stop record, so subsequent records would be ignored. This allows programming the records in several sessions, each time appending new records at the end of the list, as shown in Figure 56-2.


No Start Record
No Start Record
No Start Record
No Start Record
No Start Record
Start Record
Data Record - CS1, Ad = 0
Data Record - CS2, Ad = 0
Data Record - CS0, Ad = 0
Stop Record
Start Record
Data Record - CS0, Ad = 0
Data Record - CS1, Ad = 0
Data Record - CS2, Ad = 0
Data Record - CS0, Ad = 0
Stop Record
overwrite

Empty flash –>No action

Initial Programming

Extension

Figure 56-2. Appending DCF records

In this case, a record may overwrite a client's value which was already set by a previous record. However whether such an operation is allowed depends on the rules given for the client - some clients are write-once, only allow setting bits but not clearing, and so on.

The parity bit (PRTY) is required for some DCF records. The parity scheme used is even parity, so the number of 1s in the WDATA and PRTY fields needs to be even. (For example, if the WDATA field has the value 0000_0001h then the PRTY field needs to be set to 1, so that the total number of 1s is even.)

NOTE

When programming DCF records, ensure that programming is not interrupted and can complete without error.

56.4.2 CERS-DCF Mechanism

The figure below shows that a faulty DCF record has been loaded between the original DCF record set and the final DCF record set. When a faulty DCF record is encountered, the SSCM_STATUS[CERS] field indicates that the device configuration is not correct. In a situation like the one shown below, CERS can be cleared by writing 1 to it, as long as the final DCF record set is correct.

If there is an error in the final set of loaded DCF records, CERS cannot be cleared. This situation indicates that the final DCF record set has at least one error. To fix this, the user should update flash memory after the faulty DCF record with a correct DCF record of the same type. For example, if the faulty DCF record is DCF2, then the correct DCF record should also be DCF2.

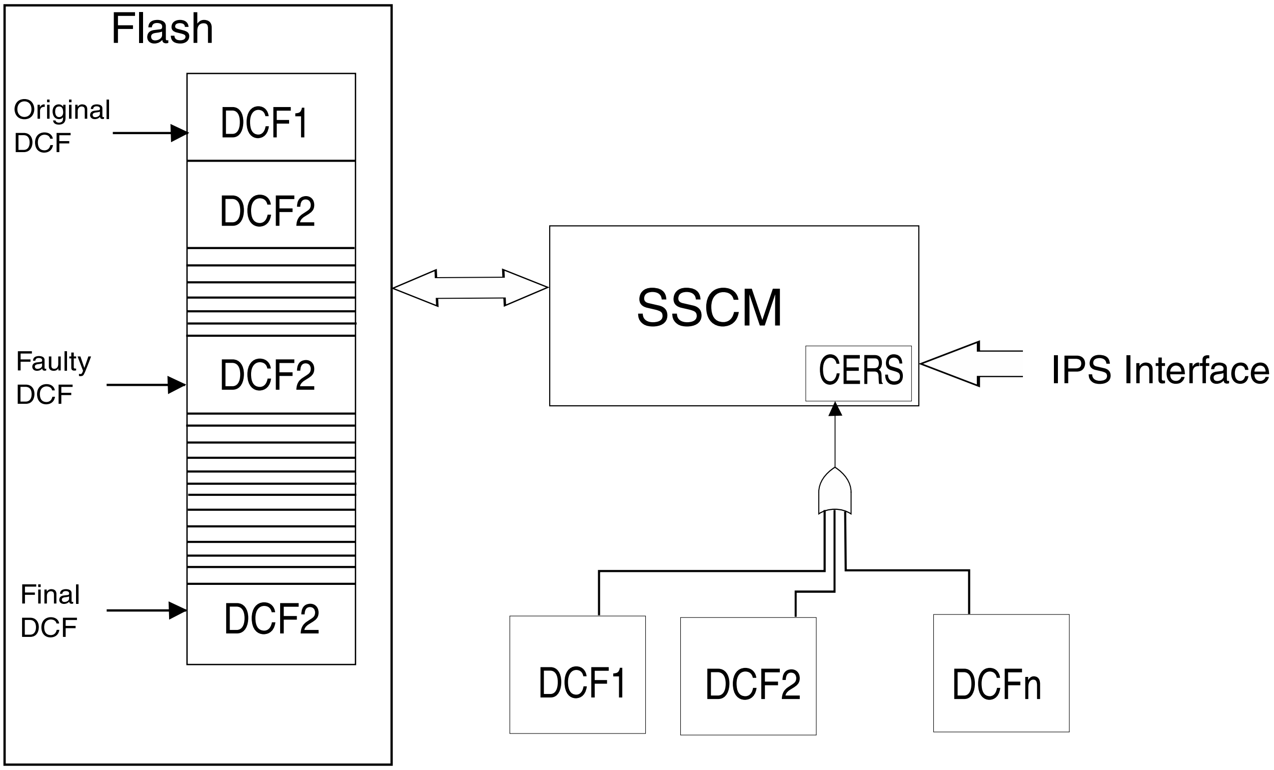


Figure 56-3. DCF Error and CERS Handling

56.4.3 Boot mode functionality

The device supports the following boot modes for the main boot core:

- Single Chip (SC) - The device will boot from the first bootable section of the flash memory main array.
- Serial Boot (SBL) - The device will download boot code from either SCI or CAN interface and then execute it.

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location) then the device will enter static mode.

56.4.4 Hardware configuration

The device will detect the boot mode based on external pins and device status. The following sequence applies:

- If the FAB (Force Alternate Boot Mode) pin is set to boot in serial mode the device can be forced into an Alternate Boot Loader Mode. The type of alternate boot mode is selected according to the ABS (Alternate Boot Selector) pins (See Table 56-7). For details of the serial boot modes please refer to the BAM/System Boot chapter.
- If the device identifies a flash memory sector with a valid boot sector, it will boot from the lowest sector. (See Figure 56-4.)
- If none of the flash memory sectors contains a valid boot signature, the device will go into static mode.

Table 56-7. Hardware Configuration

FAB	ABS 2, 0	Boot ID	Boot Mode
1	0 0	–	Serial Boot SCI
1	0 1	–	Serial Boot CAN
0	–	valid	SC (Single Chip)
0	–	not found	Static Mode

NOTE

All combinations not listed in the table above are not supported.

56.4.5 Single Chip boot sector search

56.4.5.1 Potential boot sectors

As shown in Figure 56-4 in single chip mode, several locations are searched for a valid boot ID. The lowest sector that starts with a valid boot ID is used to boot the device.

For the flash memory locations that are searched on this device, see the chip-specific details about boot locations in flash memory.

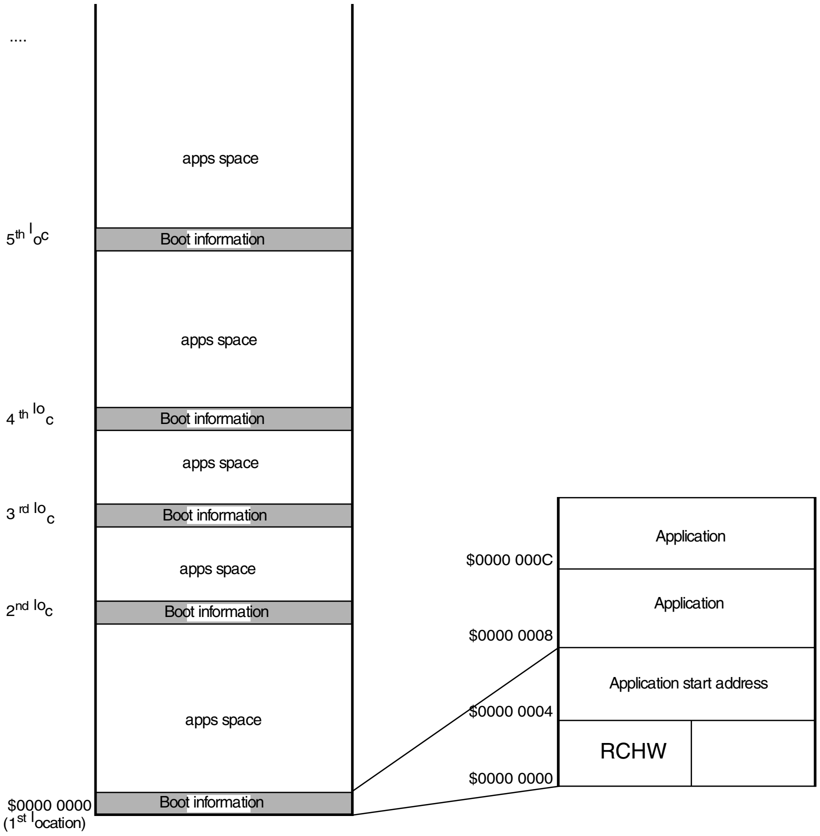


Figure 56-4. Flash memory partitioning and RCHW search

56.4.5.2 Reset Configuration Half-Word

Each boot sector in flash memory contains the Reset Configuration Half-Word (RCHW) at offset 00h. If the RCHW field BOOT_ID holds the value 5Ah then the sector is considered bootable. In addition, there is a flag which indicates that the code is VLE code. (If a device only supports VLE code for future compatibility the flag must always be set to 1.) All other bits are reserved.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	VLE	BOOT_ID							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

▨ = Reserved

Once the device detects that it needs to boot from flash memory, and finds a valid BOOT_ID, it will boot from the application start address at offset 04h within the boot sector.

NOTE

When programming any of the locations in the RCHW list, ensure that programming is not interrupted and can complete without error. When erasing any block containing any of the locations in the RCHW list, ensure that the erase is not interrupted (especially by reset or loss of power).

56.4.5.3 Boot and alternate boot sectors

Some applications require an alternate boot sector so that the main boot can be erased and reprogrammed in the field.

When an alternate boot is needed, the user can create two bootable sectors; the lowest sector shall be the main boot sector and the highest shall be the alternate boot sector. The alternate boot sector does not need to be consecutive to the main boot sector.

This scheme ensures that there is always one active boot sector by erasing one of the boot sectors only:

- Sector is activated (that is, program a valid BOOT_ID instead of FFh as initially programmed).
- Sector is deactivated by writing 0 to some bits of the BOOT_ID field (bit 1 and/or bit 3, and/or bit 4, and/or bit 6).

56.4.6 Serial boot loader mode

56.4.6.1 Serial boot loader mode - public password enabled

In Serial Boot Loader Mode, if public serial access is allowed, the MCU executes BAM code which will check for a valid public password on the chosen interface. The interface will be selected via FAB and ABS pins.

- Nexus/JTAG available
- Boot from BAM
- Flash memory main array and shadow block access disabled

56.4.6.2 Serial boot loader mode - flash memory password enabled

It's possible to boot in Serial Boot Loader Mode when serial access with password is selected. The MCU executes BAM code which will check for a valid flash memory password on the chosen interface. If the password is known, full access to the device is enabled.

- Nexus/JTAG available
- Boot from BAM
- Flash memory main array and UTEST block access possible

56.4.7 Security

The censorship mechanism implemented on the device is intended to increase the protection against unauthorized access to the device. The mechanisms employed does not provide a guarantee of a secure device. It can only be deemed effective if used in conjunction with an appropriately robust software and documentation infrastructure to assist in guarding against unauthorized access to device resources.

When the Flash is censored, the MCU is in the Secured state. This feature intends to prevent the unauthorized read and write of memory contents. Which device features are enabled or disabled is determined by the chip mode and security state in effect.

The user needs to keep in mind that part of the security must lie with the application code. As an extreme example: it would be possible to generate application code that dumps the contents of the internal program - obviously this code would defeat the purpose of security. However there may be good reasons to provide a back door in the application code. Care must be taken when implementing these backdoors, since they allow paths of attack to break security on the device.

56.4.7.1 Securing the microcontroller

The device can be secured by adding a Non-Volatile System Censorship Information (NVSCI) record to the DCF record list in the UTEST Flash block (Table 56-8).

Table 56-8. NVSCI DCF record

0:31	
Serial Boot Control	Censorship Control

0:31
0100_000Ch

A value of 55AAh in the censorship control word of the NVSCI record determines that the device is unsecured, any other value determines that the device is secured.

By factory default the device has been programmed with a DCF record in UTEST with the value 55AA_55AAh. (For the function of the Serial Boot Control word see Serial access security.)

The device supports a backdoor to unsecure the device via a 64 bit password, which is programmed in two records (Non-Volatile Password Low (NVPWDL) and Non-Volatile Password High (NVPWDH)). (Table 56-9 and Table 56-10).

Table 56-9. NVPWDH DCF record

0:31
Password High
0:31
0100_0008h

Table 56-10. NVPWDL DCF record

0:31
Password Low
0:31
0100_0004h

By default there is no backdoor password. In order to define one the two DCF records need to be added to the list.

To protect against voltage manipulations, each 16-bit halfword needs to contain both 1's and 0's. Below are examples for legal and illegal passwords:

Illegal Passwords	Legal Passwords
0000_0000_0000_0000h	0001_0010_0100_1000h
FFFF_FFFF_FFFF_FFFFh	FFFE_FFFE_FFFE_FFFEh
FFFF_0000_FFFF_FFFFh	FFF0_000F_0FFF_0FFFh
0000_0000_0000_FFFFh	1000_1000_1000_FFFEh
AAAA_AAAA_AAAA_0000h	AAAA_AAAA_AAAA_0001h

To deactivate the backdoor password ("swallowing the key") the records can be programmed to have the data value 0000_0000_0000_0000h.

However once this is done neither the manufacturer nor the user can unlock device security again, other than by triggering a factory erase mechanism. Of course, application code can still implement a different backdoor scheme if there are special requirements which are not covered by the available mechanism.

56.4.7.2 Unsecuring the microcontroller

To unsecure a secured device, the backdoor password needs to be provided. This can be done by booting in SBL mode and providing the backdoor password via the serial bootloader protocol (please refer to the BAM chapter) or via JTAG command.

If SBL mode was used in conjunction with the backdoor password, the password comparison result will only be known after a delay - this is in order to avoid brute-force attacks. The data can be downloaded into SRAM during this delay.

Note

> The scheme used here does not prohibit a malicious listener from capturing the password of a valid serial download, since the key is not in encrypted form. The user must take appropriate measures to protect against access by third parties during a valid download.

If the password is correct, the device is temporarily unsecured. In this state the a new backdoor password can be programmed by adding NVPWD records or new application data can be programmed into the flash memory.

It's also possible to unlock the device via JTAG. For this the device needs to be held in reset by asserting the external reset input. Once the flash memory has completed it's internal sequence, the JTAG register CENSOR_CTRL can be written with the password in bits 63:0 and with bit 64 set to 1. The password comparator will compare the password and unsecure the device if it's correct, if serial access with the backdoor password is allowed, and if the device hasn't swallowed the key. (Only one transition on bit 64 from 0 to 1 is allowed.) The debugger needs to wait until the device is unlocked, after that a breakpoint can be set if desired, and the debugger can release the reset. The device will remain unsecured until the next reset event occurs.

NOTE

> When external reset which is configured to generate long functional reset is applied, device will become secure until correct password is loaded by DCF again. Hence debugger will get disconnected on occurrence of long external reset.

56.4.7.3 Software unsecure

Since the security state of the device is determined solely by a set of DCF records in the UTEST block, any application may choose to implement a software unsecure method, through any interface.

56.4.8 Serial access security

If booted in SBL mode, the device can be accessed via a serial interface. It is possible to use either the public or backdoor password in SBL mode. The setting of the Serial Boot Control word in the NVSCI record determines which password will be used (see Hardware configuration). If it contains the value 55AAh, and the device is secured, the backdoor password must be used. If it contains any other value, the public password must be used.

NOTE

Any password is allowed if the device is unsecured.

Access to flash memory depends on the configuration of both the Serial Boot Control and Censorship Control fields in the NVSCI. An unsecured device always allows access to flash memory regardless of the Serial Boot Control field configuration. The flash memory will not be visible if public password access is used with a secured device. If backdoor password access is used, device security is disabled and flash memory will be visible.

The application may wish to prohibit access via the serial connection. On a secured device, this can be accomplished by programming Serial Boot Control field to 55AAh, which mandates the use of the backdoor password, or programming the NVPWD records so the password is 0000_0000_0000_0000h making the backdoor password invalid (swallowing the key) (see Table 56-10 and Table 56-9).

However, it should be carefully evaluated whether this scenario is desirable. If programmed as described above, there is no longer the possibility of allowing the manufacturer to run diagnostics on a returned device apart from erasing the flash memory. Also, it will not be possible to update application software. This does not apply if the user decides to implement an alternative backdoor scheme in software (see Software unsecure).

56.5 Initialization and application information

56.5.1 Reset

The reset state of each individual bit appears in the detailed register descriptions.

56.6 Additional safety measures

56.6.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM, but not other modules) by gating the state machines status with the expected status of the MC_RGM. If a spurious reset occurs, the SSCM will not interfere with the flash memory bus or overwrite configuration registers. However, internal SSCM status will be lost.

Chapter 57
Power Management Controller block (PMC)

57.1 Introduction

57.1.1 PMC Introduction

The power management controller (PMC) consists of two blocks: analog block and a digital block. This describes the digital block of the PMC.

The PMC block, shown below, contains all of the registers and digital logic to generate all of the enable signals, trim control bits, power on resets (PORs), and test mode logic for the analog block. The PMC digital logic also controls the PMC analog outputs that are to be sensed by the Analog-to-Digital Converter (ADC) module.

The PMC digital logic also provides the enable signals, trim bits and other registers for the temperature sensors.

Custom interfaces are provided to each of:

- The Reset Generation Module (MC_RGM) block, which receives low voltage detect (LVD) values that are used for phase transition conditions during POR.
- The Flash memory, via the SSCM (and DCF client blocks), provides trim values for use during initial POR.

:

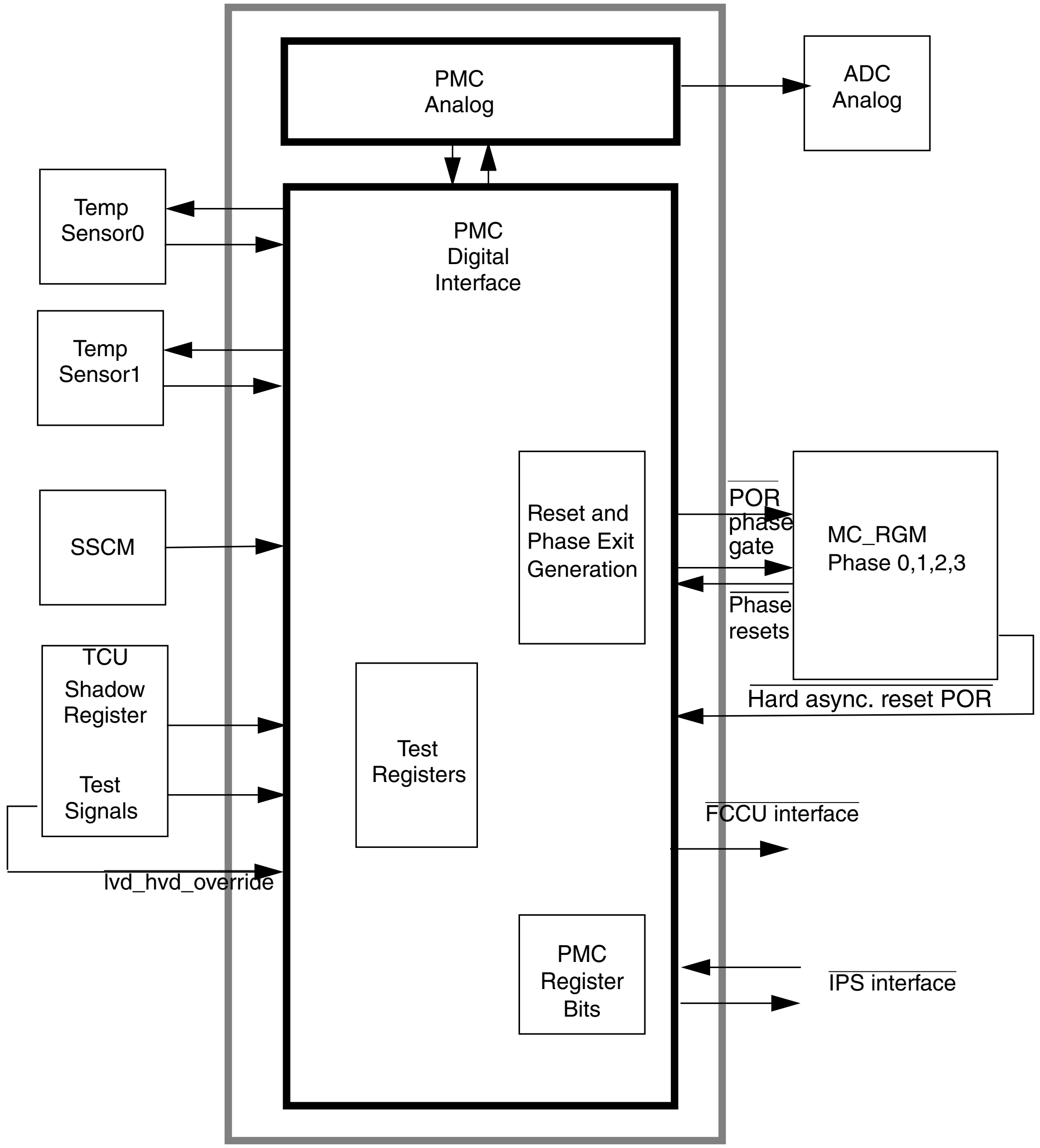


Figure 57-1. Digital PMC block diagram

57.1.2 Features

The PMC module provides:

- Configuration and control information to the analog PMC block.
- An interface with the TCU block for test related controls.

- An interface with Flash memory for retrieval of trim/control information, during POR, for HVDs, LVDs, regulators, and band gap filters.
- Necessary status and indication information to the MC_RGM block during power up.
- Configuration and monitoring of the two temperature sensors.
- Interrupt or reset generation for HVDs and LVDs (based on the configuration).
- Host access, via the IPS bus, to all register controls, configurations, and status.
- Self test logic for the LVDs and HVDs.
- Software triggered fault injection to the FCCU for LVDs, HVDs, temperature sensors, and LVD self test.

57.2 Memory Map and Registers

The PMC includes many registers for configuring, monitoring, and trimming the regulators and the LVD/HVD monitors.

PMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PMC Status Register (PMC_PMCSR)	32	R	0000_0002h	57.2.1/2244
4	Regulator Voltage Status Register (PMC_RSSR)	32	R	0000_0000h	57.2.2/2245
8	PMC Control Register (PMC_PMCCR)	32	R/W	0000_0002h	57.2.3/2248
10	Interrupt Enable Register (PMC_IER)	32	R/W	0000_0000h	57.2.4/2249
20	Event Status Register 0 (PMC_ESR_0)	32	w1c	0000_0000h	57.2.5/2253
30	Reset Event Enable 0 (PMC_REE_0)	32	R/W	0000_7C98h	57.2.6/2255
40	Reset Event Selection 0 (PMC_RES_0)	32	R/W	0000_0000h	57.2.7/2257
70	FCCU Fault Injection Register (PMC_FIR)	32	W	0000_0000h	57.2.8/2260
100	Temperature Event Status register (PMC_ESR_TD)	32	R/W	0000_0000h	57.2.9/2262
104	Temperature Reset Event Enable register (PMC_REE_TD)	32	R/W	0000_0000h	57.2.10/ 2265
108	Temperature Reset Event Selection register (PMC_RES_TD)	32	R/W	0000_0000h	57.2.11/ 2267
10C	Temperature detector configuration register (PMC_CTL_TD)	32	R/W	0000_0303h	57.2.12/ 2269
13C	LVD Self Test Time Window Register (PMC_STTW)	32	R/W	0000_007Fh	57.2.13/ 2271
140	Voltage Detect User Mode Test Register (PMC_VD_UTST)	32	R/W	0000_0000h	57.2.14/ 2272

57.2.1 PMC Status Register (PMC_PMCSR)

This status register contains the various status of the PMC block.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														INT_EXT_AUX_REG	EXT_INT_REG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

PMC_PMCSR field descriptions

Field	Description
0–29 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
30 INT_EXT_AUX_REG	External or Internal Auxiliary Regulator. This bit indicates which Aux Regulator is selected for device operation: 0 Internal Aux Regulator active 1 External Aux Regulator active
31 EXT_INT_REG	External or Internal Regulator. This bit indicates which regulator is selected for the Device operation: 0 Internal Regulator active 1 External Regulator active

57.2.2 Regulator Voltage Status Register (PMC_RSSR)

This status register contains the status of the indicated supply. This indicates the present state of the voltage detect events for each of the supplies. Each of the voltage detect signals for a give voltage are combined to form a single bit.

The following figure indicates the combined supply status generation logic for various LVDs and HVDs for a voltage level. Not that the LVD/HVD signals coming from analog is first synchronized on the system clock before generating the status.

The figure below indicates the combined supply status generation logic for various LVDs and HVDs for a voltage level. Not that the LVD/HVD signals coming from analog is first synchronized on the system clock before generating the status.

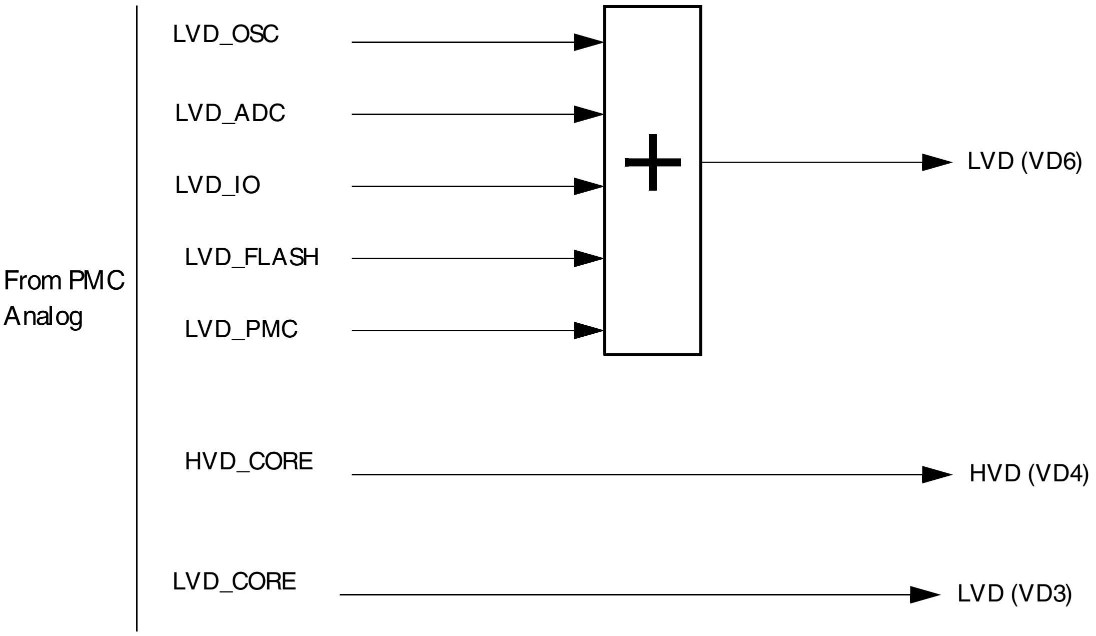


Figure 57-2. Supply Status Generation Logic

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									VD6	0	VD4	VD3	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_RSSR field descriptions

Field	Description
0–24 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
25 VD6	Voltage Detect 6 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage supplies. It is asserted when any of the supplies fall below the corresponding LVD threshold, and clears when all the supplies rise above the corresponding LVD threshold. All of the LVDs are combined to form this bit (see the figure above for details).

Table continues on the next page...

PMC_RSSR field descriptions (continued)

Field	Description
	0 Currently no occurrence. 1 LVD occurrence detected on the high voltage supplies.
26 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
27 VD4	Voltage Detect 4 high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high voltage supplies. It is asserted when any of the supplies rise above the corresponding HVD threshold, and clears when all the supplies fall below the corresponding HVD threshold. All of the HVDs are combined to form this bit (see the figure above for details). 0 Currently no occurrence. 1 HVD occurrence detected on the low voltage supplies.
28 VD3	Voltage Detect 3 low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears when the supply rises above its corresponding LVD threshold (see the figure above for details). 0 Currently no occurrence. 1 LVD occurrence detected on the low voltage point supply.
29–31 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

57.2.3 PMC Control Register (PMC_PMCCR)

This status register contains the various control settings of the PMC block.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PMCCR_EN	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	Reserved	Reserved	Reserved	Reserved	Reserved	0		Reserved	0		Reserved	Reserved	0	INT_AUX_REG_BYPASS	INT_REG_BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

PMC_PMCCR field descriptions

Field	Description
0 PMCCR_EN	PMC Control Enable. This bit determines whether any of the PMC Control bits can be written. When writing PMCCR, write 1 to PMCCR_EN to change the values of other fields in PMCCR. 0 No controls can be written. 1 Any controls can be written.
1–16 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
17 Reserved	This field is reserved. Always write as 0.
18 Reserved	This field is reserved. Always write as 0.
19 Reserved	This field is reserved. Always write as 0.
20 Reserved	This field is reserved. Always write as 0.

Table continues on the next page...

PMC_PMCCR field descriptions (continued)

Field	Description
21 Reserved	This field is reserved. Always write as 0.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved. Always write as 0.
25–26 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved. Always write as 0.
28 Reserved	This field is reserved. Always write as 0.
29 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
30 INT_AUX_REG_ BYPASS	Internal Auxiliary Regulator Bypass. This bit indicates which Aux Regulator is selected for the device's operation. **NOTE:** The Internal Auxiliary Regulator is not required for normal operation. The user can enable it if undershoots or overshoots occur when operating with the Internal Regulator. 0 Internal Aux Regulator active 1 Internal Aux Regulator disabled
31 INT_REG_ BYPASS	Internal Regulator Bypass. This bit indicates the Regulator selected for the device's operation. 0 Internal Regulator active 1 Internal Regulator disabled, External Regulator active

57.2.4 Interrupt Enable Register (PMC_IER)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Event Status Registers (ESR_0). These bits indicate whether an interrupt occurs when the voltage event is seen. A '0' indicates that no interrupt is to be sent, and a '1' indicates that an interrupt is to occur when the voltage detect event occurs. These bits are enabled via a write of one to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

NOTE

If an interrupt is enabled for an LVD/HVD, the FCCU fault corresponding to the LVD/HVD must be disabled in the FCCU.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IE_EN	0	TS1_3IE	TS1_2IE	TS1_0IE	TS0_3IE	TS0_2IE	TS0_0IE	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VD6IE_O	VD6IE_ADC	VD6IE_IM	VD6IE_F	VD6IE_C	0		VD4IE_C	0		VD3IE_H	VD3IE_C	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_IER field descriptions

Field	Description
0 IE_EN	Interrupt Enable. This bit determines whether any of the Interrupt Enable bits can be written. 0 No interrupt enables can be written. 1 Any interrupt enable can be written.
1 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
2 TS1_3IE	Temperature Sensor 1 input 3 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature rises above its corresponding threshold (flag2). 0 No interrupt occurs. 1 An interrupt occurs.
3 TS1_2IE	Temperature Sensor 1 input 2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature rises above its corresponding threshold (flag1). 0 No interrupt occurs. 1 An interrupt occurs.
4 TS1_0IE	Temperature Sensor 1 input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature falls below its corresponding threshold. 0 No interrupt occurs. 1 An interrupt occurs.
5 TS0_3IE	Temperature Sensor 0 input 3 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature rises above its corresponding threshold (flag2).

Table continues on the next page...

PMC_IER field descriptions (continued)

Field	Description
	0 No interrupt occurs. 1 An interrupt occurs.
6 TS0_2IE	Temperature Sensor 0 input 2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature rises above its corresponding threshold (flag1). 0 No interrupt occurs. 1 An interrupt occurs.
7 TS0_0IE	Temperature Sensor 0 input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature falls below its corresponding threshold. 0 No interrupt occurs. 1 An interrupt occurs.
8–16 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
17 VD6IE_O	Voltage Detect 6 Interrupt Enable Oscillator. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
18 VD6IE_ADC	Voltage Detect 6 Interrupt Enable Analog-to-Digital Converter. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
19 VD6IE_IM	Voltage Detect 6 Interrupt Enable Input-output Main. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
20 VD6IE_F	Voltage Detect 6 Interrupt Enable Flash. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
21 VD6IE_C	Voltage Detect 6 Interrupt Enable Cold. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
22–23 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMC_IER field descriptions (continued)

Field	Description
24 VD4IE_C	Voltage Detect 4 Interrupt Enable Cold. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
25–26 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
27 VD3IE_H	Voltage Detect 3 Interrupt Enable Hot This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs 1 An interrupt occurs
28 VD3IE_C	Voltage Detect 3 Interrupt Enable Cold. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
29–31 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

57.2.5 Event Status Register 0 (PMC_ESR_0)

This status register contains the latched status of the Voltage detect events. These bits indicate the state of the voltage detect events including whether a voltage level event has occurred at any time in the past but has not been cleared (via the writing '1'). These bits are read only, and they are cleared by writing a one to the corresponding ESR register.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0										Reserved	0	Reserved	0	Reserved	0
W											w1c		w1c		w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VD6_O	VD6_ADC	VD6_IM	VD6_F	VD6_C	0	Reserved	VD4_C	0		VD3_H	VD3_C	0	Reserved	Reserved
W		w1c	w1c	w1c	w1c	w1c		w1c	w1c			w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_ESR_0 field descriptions

Field	Description
0–9 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	Reserved This field is reserved.
11 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMC_ESR_0 field descriptions (continued)

Field	Description
12 Reserved	Reserved This field is reserved.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 Reserved	Reserved This field is reserved.
15–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VD6_O	Voltage Detect 6 Oscillator low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage OSC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears on writing '1'. 0 No occurrence has happened 1 LVD occurrence detected on the high voltage OSC supply.
18 VD6_ADC	Voltage Detect 6 Analog-to-Digital low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage ADC supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears on writing '1'. 0 No occurrence has happened. 1 LVD occurrence detected on the high voltage I/O ADC supply.
19 VD6_IM	Voltage Detect 6 Input-output Main low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage V I/O main supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears on writing '1'. 0 No occurrence has happened. 1 LVD occurrence detected on the high voltage I/O main supply.
20 VD6_F	Voltage Detect 6 Flash low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage flash supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears on writing '1'. 0 No occurrence has happened. 1 LVD occurrence detected on the high voltage flash supply.
21 VD6_C	Voltage Detect 6 Cold low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the high voltage cold point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears on writing '1'. 0 No occurrence has happened. 1 LVD occurrence detected on the high voltage cold point supply.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMC_ESR_0 field descriptions (continued)

Field	Description
23 Reserved	Reserved This field is reserved.
24 VD4_C	Voltage Detect 4 Cold high-voltage detect flag. This read-only bit is the high-voltage status flag associated with the voltage level detect for the high-voltage cold point supply. It is asserted when the supply rises above its corresponding HVD threshold, and clears on writing '1'. 0 No occurrence has happened. 1 HVD occurrence detected on the high-voltage supply.
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 VD3_H	Voltage Detect 3 Hot low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage hot point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears on writing '1'. 0 No occurrence has happened. 1 LVD occurrence detected on the low voltage core supply.
28 VD3_C	Voltage Detect 3 Cold low-voltage detect flag. This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage hot point supply. It is asserted when the supply falls below its corresponding LVD threshold, and clears on writing '1'. 0 No occurrence has happened. 1 LVD occurrence detected on the low voltage core supply.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	Reserved This field is reserved.
31 Reserved	Reserved This field is reserved.

57.2.6 Reset Event Enable 0 (PMC_REE_0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event.

The register's reset value applies until the portion of the boot sequence that loads a value for the register from a DCF record in the UTEST area of flash memory. The user can change the loaded value by programming a new DCF record in the UTEST area. If the loaded value programs any flag bit to be enabled (set to 1), the bit cannot be disabled (cleared to 0) by software through IPS programming.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VD6RE_O	VD6RE_ADC	VD6RE_IM	VD6RE_F	VD6RE_C	0		VD4RE_C	0		VD3RE_H	VD3RE_C	0		
W																
Reset	0	1	1	1	1	1	0	0	1	0	0	1	1	0	0	0

PMC_REE_0 field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VD6RE_O	Voltage Detect 6 Reset Enable Oscillator. This bit determines whether an reset is seen by the system when the voltage detect event occurs. 0 No reset occurs. 1 An reset occurs.
18 VD6RE_ADC	Voltage Detect 6 Reset Enable Analog-to-Digital Converter. This bit determines whether an reset is seen by the system when the voltage detect event occurs. 0 No reset occurs. 1 An reset occurs.
19 VD6RE_IM	Voltage Detect 6 Reset Enable Input-output Main. This bit determines whether an reset is seen by the system when the voltage detect event occurs. 0 No reset occurs. 1 An reset occurs.
20 VD6RE_F	Voltage Detect 6 Reset Enable Flash. This bit determines whether an reset is seen by the system when the voltage detect event occurs. 0 No reset occurs. 1 An reset occurs.

Table continues on the next page...

PMC_REE_0 field descriptions (continued)

Field	Description
21 VD6RE_C	Voltage Detect 6 Reset Enable Cold (PMU supply: 3.3 V) This bit determines whether an reset is seen by the system when the voltage detect event occurs. 0 No reset occurs. 1 An reset occurs.
22–23 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
24 VD4RE_C	Voltage Detect 4 Reset Enable Cold (Core HVD: 1.25 V). This bit determines whether an reset is seen by the system when the voltage detect event occurs. 0 No reset occurs. 1 An reset occurs.
25–26 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
27 VD3RE_H	Voltage Detect 3 Reset Enable Hot (Core LVD: 1.25 V). This bit determines whether an reset is seen by the system when the voltage detect event occurs. 0 No reset occurs. 1 An reset occurs.
28 VD3RE_C	Voltage Detect 3 Reset Enable Cold (Core LVD: 1.25 V). This bit determines whether an reset is seen by the system when the voltage detect event occurs. 0 No reset occurs. 1 An reset occurs.
29–31 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

57.2.7 Reset Event Selection 0 (PMC_RES_0)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event.

NOTE

If a functional reset is enabled for an LVD/HVD, the FCCU fault corresponding to the LVD/HVD must be disabled in the FCCU. The user should configure a functional reset for an

LVD/HVD only for debugging purposes and never for any actual application.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VD6RES_O	VD6RES_ADC	VD6RES_IM	VD6RES_F	VD6RES_C	0		VD4RES_C	0		VD3RES_H	VD3RES_C	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_RES_0 field descriptions

Field	Description
0–16 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
17 VD6RES_O	Voltage Detect 6 Reset Event Select Oscillator. This bit determines whether a functional reset or destructive reset is generated when the voltage detect event occurs. 0 Destructive reset generated. 1 Functional reset generated.
18 VD6RES_ADC	Voltage Detect 6 Reset Event Select Analog-to-Digital Converter. This bit determines whether a functional reset or destructive reset is generated when the voltage detect event occurs. 0 Destructive reset generated. 1 Functional reset generated.
19 VD6RES_IM	Voltage Detect 6 Reset Event Select Input-output Main. This bit determines whether a functional reset or destructive reset is generated when the voltage detect event occurs. 0 Destructive reset generated. 1 Functional reset generated.
20 VD6RES_F	Voltage Detect 6 Reset Event Select Flash. This bit determines whether a functional reset or destructive reset is generated when the voltage detect event occurs.

Table continues on the next page...

PMC_RES_0 field descriptions (continued)

Field	Description
	0 Destructive reset generated. 1 Functional reset generated.
21 VD6RES_C	Voltage Detect 6 Reset Event Select Cold. This bit determines whether a functional reset or destructive reset is generated when the voltage detect event occurs. 0 Destructive reset generated. 1 Functional reset generated.
22–23 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
24 VD4RES_C	Voltage Detect 4 Reset Event Select Cold. This bit determines whether a functional reset or destructive reset is generated when the voltage detect event occurs. 0 Destructive reset generated. 1 Functional reset generated.
25–26 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
27 VD3RES_H	Voltage Detect 3 Reset Event Select Hot. This bit determines whether a functional reset or destructive reset is generated when the voltage detect event occurs. 0 Destructive reset generated. 1 Functional reset generated.
28 VD3RES_C	Voltage Detect 3 Reset Event Select Cold. This bit determines whether a functional reset or destructive reset is generated when the voltage detect event occurs. 0 Destructive reset generated. 1 Functional reset generated.
29–31 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

57.2.8 FCCU Fault Injection Register (PMC_FIR)

This FCCU Fault Injection register enables the user s/w to artificially generate FCCU faults corresponding to various Events, namely LVD, HVD, TEMP SENS and Self Test. A write '1' to these bits generates a single cycle pulsed fault to the FCCU block, this bit automatically gets cleared in the next cycle.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W													STEST_Fault	TSNS_Fault	HVD_Fault	LVD_Fault
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_FIR field descriptions

Field	Description
0–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28 STEST_Fault	Self Test Fault injection. A write to this bit sends a FCCU fault (single cycle) for LVD SELF TEST 0 No fault is sent 1 Single cycle fault injected.

Table continues on the next page...

PMC_FIR field descriptions (continued)

Field	Description
29 TSNS_Fault	Temperature Sensor Fault injection. A write to this bit sends a FCCU fault (single cycle) for TEMP SENSOR. 0 No fault is sent. 1 Single cycle fault injected.
30 HVD_Fault	High Voltage Detect Fault injection. A write to this bit sends a FCCU fault (single cycle) for HVD 0 No fault is sent 1 Single cycle fault injected.
31 LVD_Fault	Low Voltage Detect Fault injection. A write to this bit sends a FCCU fault (single cycle) for LVD 0 No fault is sent 1 Single cycle fault injected.

57.2.9 Temperature Event Status register (PMC_ESR_TD)

This Event Status Register indicates the present and past state of the two temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				TEMP1_3_OP	TEMP1_2_OP	0	TEMP1_0_OP	0				TEMP0_3_OP	TEMP0_2_OP	0	TEMP0_0_OP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				TEMP1_3	TEMP1_2	0	TEMP1_0	0				TEMP0_3	TEMP0_2	0	TEMP0_0
W					w1c	w1c		w1c					w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_ESR_TD field descriptions

Field	Description
0–3 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
4 TEMP1_3_OP	Temperature Sensor 1 Output 3 This bit reflects the output of the temperature for the temperature sensor 1 point (165°C). If IER[TS1_3IE] is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_3] is also asserted, a system reset is generated, which clears IER[TS1_3IE] and negates the interrupt request. If REE_TD[TEMP1_3] is asserted and a destructive reset is selected because RES_TD[TEMP1_3] is 0, then a destructive reset is generated; if RES_TD[TEMP1_3] is 1, then a functional reset is generated.
5 TEMP1_2_OP	Temperature Sensor 1 Output 2 This bit reflects the output of the temperature for the temperature sensor 1 point (150°C). If IER[TS1_2IE] is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_2] is also asserted, a system reset is generated, which clears IER[TS1_2IE] and negates the interrupt request. If REE_TD[TEMP1_2] is asserted and a destructive reset is selected because RES_TD[TEMP1_2] is 0, then a destructive reset is generated; if RES_TD[TEMP1_2] is 1, then a functional reset is generated.
6 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
7 TEMP1_0_OP	Temperature Sensor 1 Output 0 This bit reflects the output of the temperature for the cold temperature sensor 1 point. If IER[TS1_0IE] is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_0] is also asserted, a system reset is generated, which clears IER[TS1_0IE] and negates the interrupt request. If REE_TD[TEMP1_0] is asserted and a destructive reset is selected because RES_TD[TEMP1_0] is 0, then a destructive reset is generated; if RES_TD[TEMP1_0] is 1, then a functional reset is generated.
8–11 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
12 TEMP0_3_OP	Temperature Sensor 0 Output 3 This bit reflects the output of the temperature for the temperature sensor point (165°C). If IER[TS0_3IE] is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP0_3] is also asserted, a system reset is generated, which clears IER[TS0_3IE] and negates the interrupt request. If REE_TD[TEMP0_3] is asserted and a destructive reset is selected because RES_TD[TEMP0_3] is 0, then a destructive reset is generated; if RES_TD[TEMP0_3] is 1, then a functional reset is generated.
13 TEMP0_2_OP	Temperature Sensor 0 Output 2 This bit reflects the output of the temperature for the temperature sensor point (150°C). If IER[TS0_2IE] is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP0_2] is also asserted, a system reset is generated, which clears IER[TS0_2IE] and negates the interrupt request. If REE_TD[TEMP0_2] is asserted and a destructive reset is selected because RES_TD[TEMP0_2] is 0, then a destructive reset is generated; if RES_TD[TEMP0_2] is 1, then a functional reset is generated.
14 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMC_ESR_TD field descriptions (continued)

Field	Description
15 TEMP0_0_OP	Temperature Sensor 0 Output 0 This bit reflects the output of the temperature for the cold temperature sensor point. If IER[TS0_0IE] is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP0_0] is also asserted, a system reset is generated, which clears IER[TS0_0IE] and negates the interrupt request. If REE_TD[TEMP0_0] is asserted and a destructive reset is selected because RES_TD[TEMP0_0] is 0, then a destructive reset is generated; if RES_TD[TEMP0_0] is 1, then a functional reset is generated.
16–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20 TEMP1_3	Temperature sensor 1 status flag 3. This bit is the temperature status flag associated with the temperature for the high temperature sensor 1 point (165C). Its value becomes 1 when the temperature exceeds its corresponding threshold, and that value clears to 0 when 1 is written to this location. 0 Currently no occurrence. 1 Temperature occurrence detected.
21 TEMP1_2	Temperature sensor 1 status flag 2. This bit is the temperature status flag associated with the temperature for the temperature sensor 1 point (150C). Its value becomes 1 when the temperature exceeds its corresponding threshold, and that value clears to 0 when 1 is written to this location. 0 Currently no occurrence. 1 Temperature occurrence detected.
22 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
23 TEMP1_0	Temperature sensor 1 status flag 0. This bit is the temperature status flag associated with the temperature for the cold temperature sensor 1 point. Its value becomes 1 when the temperature exceeds its corresponding threshold, and that value clears to 0 when 1 is written to this location. 0 Currently no occurrence. 1 Temperature occurrence detected.
24–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28 TEMP0_3	Temperature sensor 0 status flag 3. This bit is the temperature status flag associated with the temperature for the high temperature sensor point (165C). Its value becomes 1 when the temperature exceeds its corresponding threshold, and that value clears to 0 when 1 is written to this location. 0 Currently no occurrence. 1 Temperature occurrence detected.
29 TEMP0_2	Temperature sensor 0 status flag 2.

Table continues on the next page...

PMC_ESR_TD field descriptions (continued)

Field	Description
	This bit is the temperature status flag associated with the temperature for the temperature sensor point (150C). Its value becomes 1 when the temperature exceeds its corresponding threshold, and that value clears to 0 when 1 is written to this location. 0 Currently no occurrence. 1 Temperature occurrence detected.
30 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
31 TEMP0_0	Temperature sensor 0 status flag 0. This bit is the temperature status flag associated with the temperature for the cold temperature sensor point. Its value becomes 1 when the temperature exceeds its corresponding threshold, and that value clears to 0 when 1 is written to this location. 0 Currently no occurrence. 1 Temperature occurrence detected.

57.2.10 Temperature Reset Event Enable register (PMC_REE_TD)

This Reset Event Enable register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

The register's reset value applies until the portion of the boot sequence that loads a value for the register from a DCF record in the UTEST area of flash memory. The DCF record for this register is PMC_LVD_MISC. The user can change the loaded value by programming a new DCF record for PMC_LVD_MISC in the UTEST area. If the loaded value programs any flag bit to be enabled (set to 1), the bit cannot be disabled (cleared to 0) by software through IPS programming.

NOTE

Programming a new value for PMC_LVD_MISC in the UTEST area of flash memory requires special care. In addition to the TEMPn_m bits, this DCF record contains the LVD trim values for the ADC, FLASH, IO, and OSC 3.3 V supplies. For these LVD trim bits, the user must not change the values in the factory-programmed DCF record for PMC_LVD_MISC.

1. Before programming a new value for PMC_LVD_MISC, read the factory-programmed value of the DCF record.

2. When programming a new DCF record for PMC_LVD_MISC:
   - Change only the values of the desired TEMPn_m bits.
   - Ensure that all other bits have the same values as in the factory-programmed value of PMC_LVD_MISC.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				TEMP1_3	TEMP1_2	0	TEMP1_0	0				TEMP0_3	TEMP0_2	0	TEMP0_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_REE_TD field descriptions

Field	Description
0–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20 TEMP1_3	Temperature sensor 1 detect assertion reset enable 3. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP1_3] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
21 TEMP1_2	Temperature sensor 1 detect assertion reset enable 2. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP1_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
22 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
23 TEMP1_0	Temperature sensor 1 detect assertion reset enable 0. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP1_0] bit determines whether the reset is functional or destructive.

Table continues on the next page...

PMC_REE_TD field descriptions (continued)

Field	Description
	0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
24–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28 TEMP0_3	Temperature sensor 0 detect assertion reset enable 3. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP0_3] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
29 TEMP0_2	Temperature sensor 0 detect assertion reset enable 2. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP0_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
30 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
31 TEMP0_0	Temperature sensor 0 detect assertion reset enable 0. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP0_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.

57.2.11 Temperature Reset Event Selection register (PMC_RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the temperature sensor event causes either a destructive or functional reset to be generated when the selected temperature passes the trigger event.

The register's reset value applies until the portion of the boot sequence that loads a value for the register from a DCF record in the UTEST area of flash memory. The user can change the loaded value by programming a new DCF record in the UTEST area.

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				TEMP1_3	TEMP1_2	0	TEMP1_0	0				TEMP0_3	TEMP0_2	0	TEMP0_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_RES_TD field descriptions

Field	Description
0–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20 TEMP1_3	TEMP1_3 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
21 TEMP1_2	TEMP1_2 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
22 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
23 TEMP1_0	TEMP1_0 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
24–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28 TEMP0_3	TEMP0_3 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset.

Table continues on the next page...

PMC_RES_TD field descriptions (continued)

Field	Description
	0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
29 TEMP0_2	TEMP0_2 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
30 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
31 TEMP0_0	TEMP0_0 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.

57.2.12 Temperature detector configuration register (PMC_CTL_TD)

NOTE

To disable a temperature sensor, program both TSn_DOUT_EN and TSn_AOUT_EN for that temperature sensor to 0. To enable a temperature sensor, write 1 to either or both of these fields.

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TS1_ADJ_TEMP_SENS[4:0]					TS1_DOUT_EN	TS1_AOUT_EN	0	TS0_ADJ_TEMP_SENS[4:0]					TS0_DOUT_EN	TS0_AOUT_EN
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

PMC_CTL_TD field descriptions

Field	Description
0–16 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
17–21 TS1_ADJ_ TEMP_ SENS[4:0]	Customer-adjustable trim bits for under temperature 1 (cold) and over temperature 1 (hot) threshold detection flags. These bits are a signed binary number that is either added to or subtracted from the trim values of the Temperature Sensor 1 threshold detection flags. The maximum value to add or subtract is 7. TS1_ADJ_TEMP_SENS[4] is a signed bit. • When TS1_ADJ_TEMP_SENS[4] is 0, TS1_ADJ_TEMP_SENS[3] must be written as 0, and the value represented by TS1_ADJ_TEMP_SENS[2:0] is added to the trim values of the temperature threshold detection flags. • When TS1_ADJ_TEMP_SENS[4] is 1, the value in two's-complement format represented by TS1_ADJ_TEMP_SENS[3:0] is subtracted from the trim values of the temperature threshold detection flags. **NOTE:** This example shows how to program a value for subtraction in two's-complement format. If the subtrahend value is the maximum value, 0111: • The value's one's-complement form is 1000. • The value's two's-complement form is 1001. • To subtract the value, program TS1_ADJ_TEMP_SENS[4:0] with 11001.
22 TS1_DOUT_EN	Temperature Sensor 1 Digital Output Enable. 0 Disable digital output. 1 Enable digital output.
23 TS1_AOUT_EN	Temperature Sensor 1 Analog Output Enable. Analog Output Enable. 0 Disable analog output. 1 Enable analog output.
24 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
25–29 TS0_ADJ_ TEMP_ SENS[4:0]	Temperature Sensor 0 Trim Adjust Value Customer-adjustable trim bits for under temperature 0 (cold) and over temperature 0 (hot) threshold detection flags. These bits are a signed binary number that is either added to or subtracted from the trim values of the Temperature Sensor 0 threshold detection flags. The maximum value to add or subtract is 7. TS0_ADJ_TEMP_SENS[4] is a signed bit. • When TS0_ADJ_TEMP_SENS[4] is 0, TS0_ADJ_TEMP_SENS[3] must be written as 0, and the value represented by TS0_ADJ_TEMP_SENS[2:0] is added to the trim values of the temperature threshold detection flags. • When TS0_ADJ_TEMP_SENS[4] is 1, the value in two's-complement format represented by TS0_ADJ_TEMP_SENS[3:0] is subtracted from the trim values of the temperature threshold detection flags. **NOTE:** This example shows how to program a value for subtraction in two's-complement format. If the subtrahend value is the maximum value, 0111:

Table continues on the next page...

PMC_CTL_TD field descriptions (continued)

Field	Description
	• The value's one's-complement form is 1000. • The value's two's-complement form is 1001. • To subtract the value, program TS0_ADJ_TEMP_SENS[4:0] with 11001.
30 TS0_DOUT_EN	Temperature Sensor 0 Digital Output Enable. 0 Disable digital output. 1 Enable digital output.
31 TS0_AOUT_EN	Temperature Sensor 0 Analog Output Enable. 0 Disable analog output. 1 Enable analog output.

57.2.13 LVD Self Test Time Window Register (PMC_STTW)

This register configures the Self test Time window for each LVD.

Address: 0h base + 13Ch offset = 13Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																				STTW											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

PMC_STTW field descriptions

Field	Description
0–19 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
20–31 STTW	Self Test Time Window. These bits are used to control the Time window for the self test for each LVD. This is based on the clock frequency used. E.g., for 8us window, with 100 MHZ clock, this should be configured with 800 decimal value.

57.2.14 Voltage Detect User Mode Test Register (PMC_VD_UTST)

This register is used for testing of the LVDs and HVDs during user mode.

Address: 0h base + 140h offset = 140h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														ST_RESULT	ST_DONE
W															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								ST_MODE		VD_ST_CTRL[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_VD_UTST field descriptions

Field	Description
0–13 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
14 ST_RESULT	Self Test Result. Self Test result status. (Valid only when ST_DONE is HIgh). It is cleared on reset or writing ‘1’ by the host.

Table continues on the next page...

PMC_VD_UTST field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>0 Self Test Failed
1 Self Test Passed</td></tr>
<tr><td>15
ST_DONE</td><td>Self Test Done status.
Self Test completed indication status.
0 Self test running
1 Self test completed.</td></tr>
<tr><td>16–23
Reserved</td><td>Reserved
This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>24–25
ST_MODE</td><td>Self Test mode bits for testing of LVDs and HVDs.
00 Default
01 S/W triggered self test.
10 Single VD test as indicated by the VD_UTST bits.
11 Reserved</td></tr>
<tr><td>26–31
VD_ST_
CTRL[5:0]</td><td>Voltage Detect Self Test Control.
User mode Test bits for testing of LVDs and HVDs.
<table>
<tr><th>VD_ST_CTRL[5:0]</th><th>Test</th></tr>
<tr><td>000000</td><td>Functional Mode</td></tr>
<tr><td>000001</td><td>Reserved</td></tr>
<tr><td>000010</td><td>Reserved</td></tr>
<tr><td>000011</td><td>LVD_CORE</td></tr>
<tr><td>000100</td><td>HVD_CORE</td></tr>
<tr><td>000101</td><td>Reserved</td></tr>
<tr><td>000110</td><td>LVD_VDDREG</td></tr>
<tr><td>000111</td><td>LVD_FLASH</td></tr>
<tr><td>001000</td><td>LVD_ADC</td></tr>
<tr><td>001001</td><td>LVD_OSC</td></tr>
<tr><td>001010</td><td>LVD_IO</td></tr>
<tr><td>001011</td><td>Reserved</td></tr>
<tr><td>001100</td><td>Reserved</td></tr>
<tr><td>001101</td><td>LVD_CORE_HOT</td></tr>
<tr><td>001110</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

57.3 Analog PMC interface

The digital block uses the analog PMC interface to configure trim values for the regulators and LVDs that are in the analog block.

57.4 Temperature Sensor Interface Logic

The digital PMC block provides control and configuration information to the two temperature sensors. This information includes digital and analog enables, and trim values. A user adjustable trim control is also provided to fine tune the trimming (for more detail, see the detailed description of the Temperature Sensor module).

57.5 Power On Reset (MC_RGM Phase Gates)

The interface with the MC_RGM block is used during the power on sequence and startup. It provides various phase gates, thereby enabling the MC_RGM to properly complete the reset sequence.

It also provides functional and destructive reset sources, based on the configuration. Two types of resets are generated, based on LVD and HVD faults and the corresponding REE and RES enables:

- functional, and
- destructive.

The temperature sensor provides another reset source, based on the under/over temperature faults and the corresponding REE and RES enables.

57.6 Flash interface (DCF)

During power up, trim values for the regulators, LVDs, and HVDs are read from Flash memory by the SSCM, and are then provided to the digital PMC via the DCF interface.

The reset event enable (REE) configurations that are stored in Flash memory are also provided to the PMC via the DCF interface. These REEs configurations are used, by default, to generate various resets to the RGM.

57.7 FCCU Interface

The digital block generates various fault indications to the FCCU block:

- Separate corresponding, active-high, fault indications are generated whenever an LVD or HVD defect is detected in the analog block.
- A fault indication is generated when a self test task results in a failure.
- A fault condition is generated whenever over-temperature or under-temperature events occur in the temperature sensor.

The figures below show the exact logic for the generation fault indications to FCCU block.

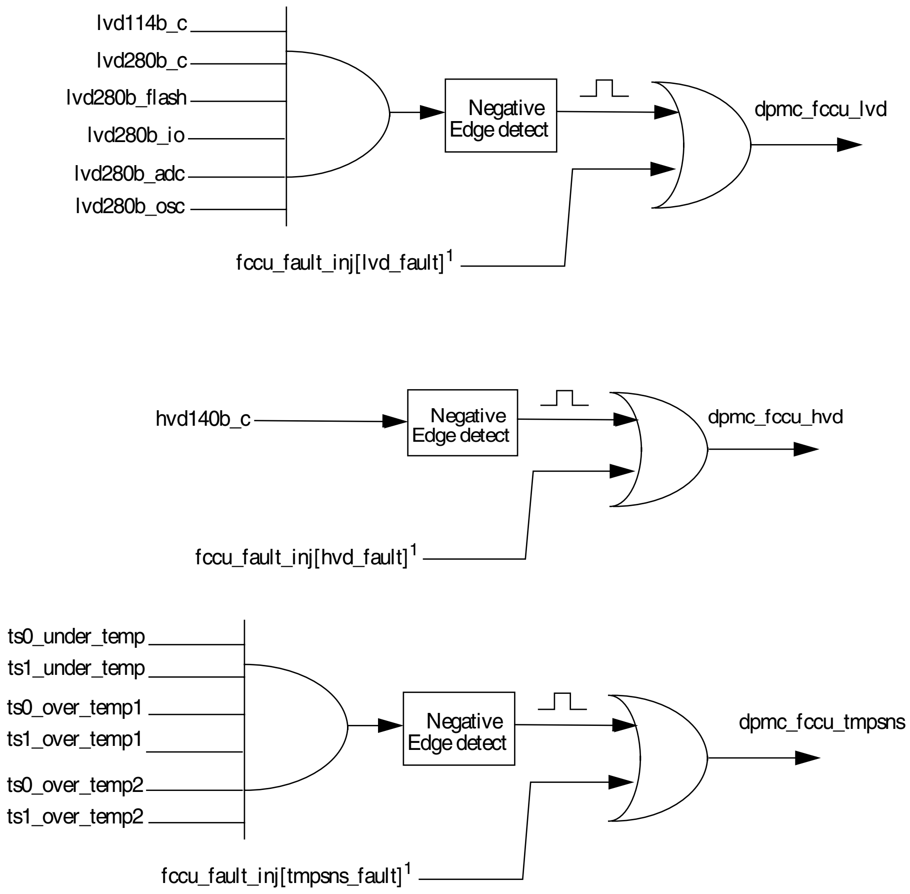


Figure 57-3. Fault generation to FCCU

:

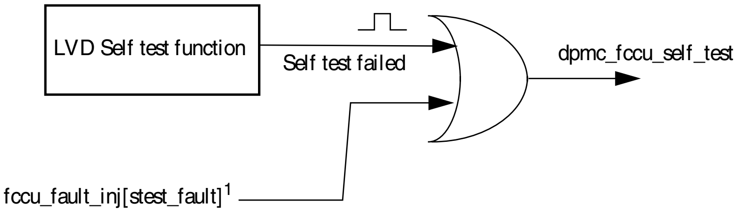


Figure 57-4. FCCU_fault_gen_LVD_Self_Test

[1] See “FCCU Fault Injection Register 0(FIR) for details.

57.8 LVD/HVD self test mechanism

The PMC implements an LVD/HVD self test mechanism for use in satisfying safety goals. The digital block assists and controls the analog block during it's performance of the self test.

Currently, three self test modes (described in the following sections) are supported by the analog block (based on the mode specified by the VD_UTST[ST_MODE] register bit).

The self test status and result are provided in the VD_UTST[ST_DONE, ST_RESULT] register fields, respectively.

57.8.1 Default Mode

The self test is always performed during startup (power on). The self test starts during the phase 3 MC_RGM state, when the SSCM has completed reads from the flash memory (that is, asserts ‘sscm_done’ signal). A counter is provided that starts from zero and increments by one after an 8 µs time gap.

Meanwhile, the LVD/HVD indication from the analog block is latched in the VD_UTST register. At the end of the self test (that is, when all of the LVDs/HVDs have been tested), the VD_UTST[ST_DONE] status bit is set to '1' and all of the bits in the VD_UTST register are checked for '1'. One or more bits not set to '1' generates a fault indication to the FCCU block, and the VD_UTST[ST_RESULT] register bit is set to '0'. In the case where all of the register bits are set to '1', the test is considered to be PASSED, the VD_UTST[ST_RESULT] register bit is set to '1', and no fault indications are provided to the FCCU.

NOTE

The total run time of the LVD self test is equal to: 'time required per LVD (8 µs)' multiplied by 'number of LVDs (14)' = 112 µs. Some of the 14 self tests are internal self tests that are listed as reserved in the description of VD_UTST[VD_ST_CTRL].

57.8.2 Software triggered self test

This mode is entered by software by writing '01' to the VD_UTST[ST_MODE] register 2-bit field. The complete self test, as is performed in default mode, is repeated. The difference is that the self test in this mode is initiated by the host while the self test in default mode is started automatically during power on.

NOTE

The LVD self test Time Window Register (STTW) has to be configured appropriately (based on the clock frequency used), before initiating software triggered self test. For example, the configuration for a 100MHz system clock and a 8us window should be 800 (decimal).

57.8.3 Single VD test

This mode is entered by software writing 10b to the 2-bit VD_UTST[ST_MODE] field, which results in the assertion of the pmc_lvd_self_test input. The test completes after the prescribed time gap has elapsed. The PASS/FAIL result is then updated based on whether or not the corresponding LVD output trips.

NOTE

The LVD Self Test Time Window Register (STTW) must to be configured appropriately as in Software triggered self test mode.

NOTE

During the running of the LVD self test, all of the LVD and HVD outputs are masked that are also under self test (see the LVD/HVD list in the definition of VD_UTST[VD_ST_CTRL]). Therefore, they have no effect on POR generation, or in any reset and/or fault generation.

Chapter 58
Power Control Unit (MC_PCU)

58.1 Introduction

58.1.1 Overview

The power control unit (MC_PCU) acts as a bridge for mapping the PMC peripheral to the MC_PCU address space.

The following figure depicts the MC_PCU block diagram.

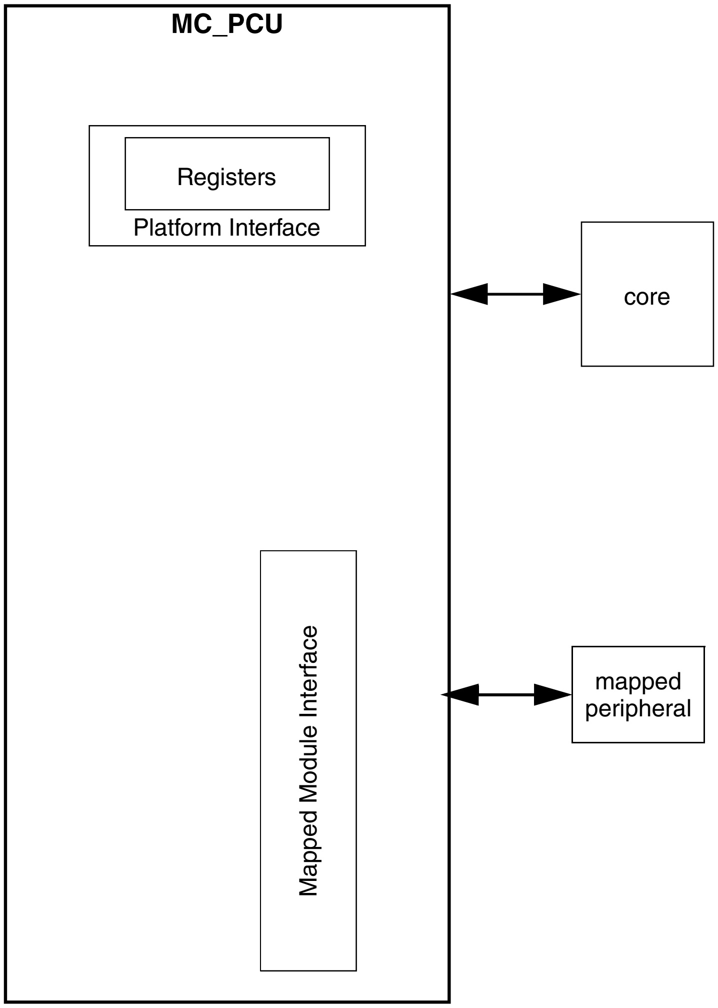


Figure 58-1. MC_PCU Block Diagram

58.1.2 Features

The MC_PCU includes the following features:

- maps the PMC registers to the MC_PCU address space

58.2 External Signal Description

The MC_PCU has no connections to any external pins.

58.3 Memory Map and Register Definition

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the PD0 field of the PCU_PSTAT register may be accessed as a word at address 0x0040, as a half-word at address 0x0042, or as a byte at address 0x0043.

MC_PCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Power Domain #0 Configuration Register (MC_PCU_PCONF0)	32	R	0000_05FFh	58.3.1/2284
40	Power Domain Status Register (MC_PCU_PSTAT)	32	R	0000_0001h	58.3.2/2286

58.3.1 Power Domain #0 Configuration Register (MC_PCU_PCONF0)

This register defines for power domain #0 whether it is on or off in each chip mode. It can be read in user and supervisor mode. Because power domain #0 is always on, none of this register's bits are programmable. This register is available for completeness reasons.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

MC_PCU_PCONF0 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	Power domain control during STOP0 mode 0 Power domain off 1 Power domain on
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HALT0	Power domain control during HALT0 mode 0 Power domain off 1 Power domain on

Table continues on the next page...

MC_PCU_PCONF0 field descriptions (continued)

Field	Description
24 RUN3	Power domain control during RUN3 mode 0 Power domain off 1 Power domain on
25 RUN2	Power domain control during RUN2 mode 0 Power domain off 1 Power domain on
26 RUN1	Power domain control during RUN1 mode 0 Power domain off 1 Power domain on
27 RUN0	Power domain control during RUN0 mode 0 Power domain off 1 Power domain on
28 DRUN	Power domain control during DRUN mode 0 Power domain off 1 Power domain on
29 SAFE	Power domain control during SAFE mode 0 Power domain off 1 Power domain on
30 TEST	Power domain control during TEST mode 0 Power domain off 1 Power domain on
31 RST	Power domain control during RESET mode 0 Power domain off 1 Power domain on

58.3.2 Power Domain Status Register (MC_PCU_PSTAT)

This register reflects the power status of all available power domains. It can be read in user and supervisor mode.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															PD0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MC_PCU_PSTAT field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PD0	Power status for power domain #n 0 Power domain is inoperable 1 Power domain is operable

Chapter 59
Mode Entry Module (MC_ME)

59.1 Introduction

59.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

The following figure shows the MC_ME block diagram.

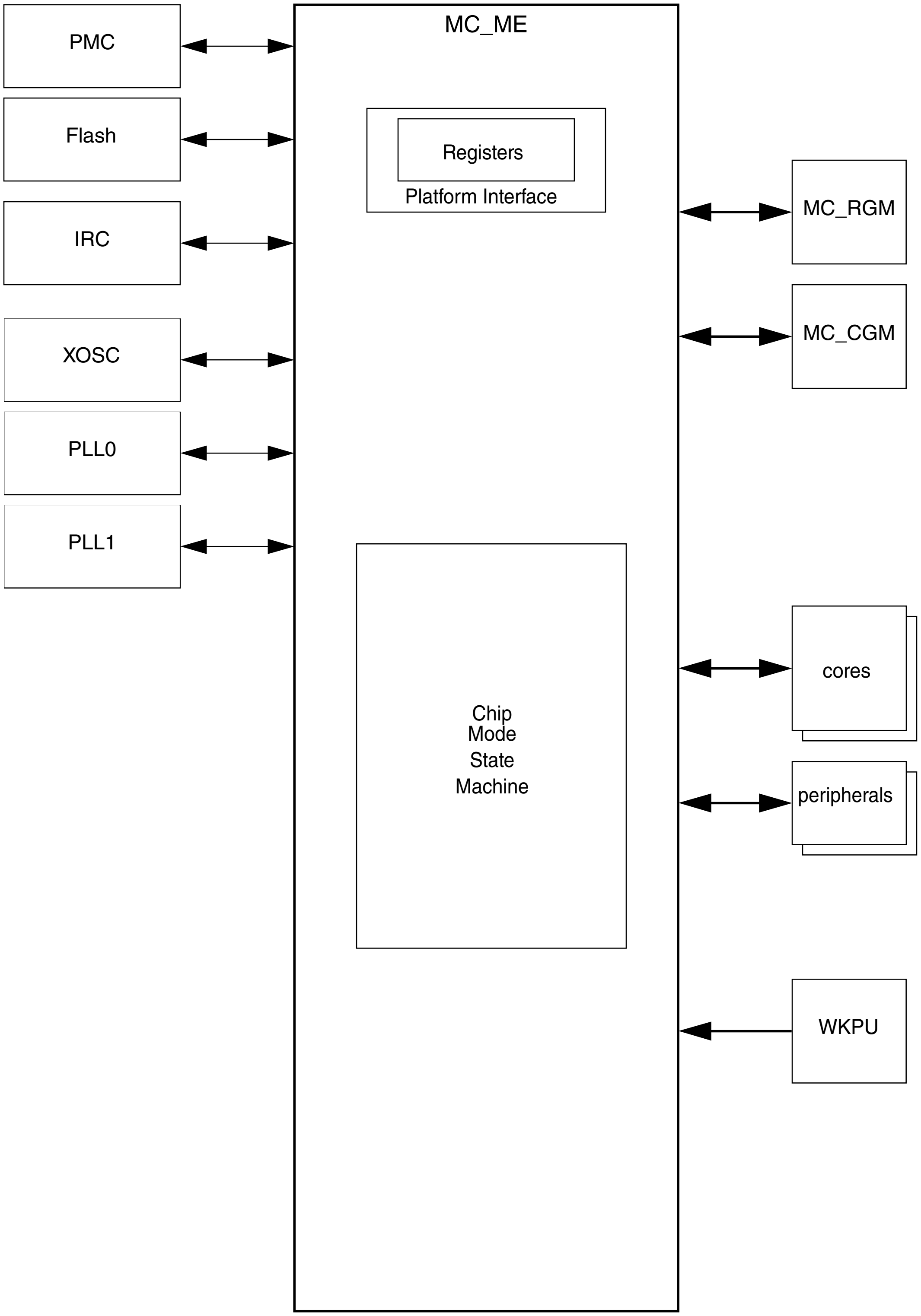


Figure 59-1. MC_ME Block Diagram

59.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the ME_ME register
- definition of various chip mode configurations by the ME_<mode>_MC registers
- control of the actual chip mode by the ME_MCTL register
- capture of the current mode and various resource status within the contents of the ME_GS register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTLn registers
- additional core clock gating and boot address control based on the ME_CCTLn and ME_CADDR***n*** registers
- capture of current peripheral and additional core clock gated/enabled status
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

59.1.3 Modes of Operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as RESET, DRUN, SAFE, and TEST. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as RUN0…3, HALT0, and STOP0 which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, SAFE, TEST, and RUN0…3 are the chip software running modes.

The following table describes the MC_ME modes.

Table 59-1. MC_ME Mode Descriptions

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAM when present is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from SAFE, TEST and RUN0…3	system reset assertion, RUN0…3, TEST via software, SAFE via software or hardware failure.
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN, TEST, and RUN0…3	system reset assertion, DRUN via software
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from DRUN	system reset assertion, DRUN via software
RUN0…3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from DRUN or other RUN0…3, interrupt event from HALT0, interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, DRUN and other RUN0…3 modes, HALT0, STOP0 via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals like clock sources, flash, main regulator, etc. for efficient power management at the cost of higher wakeup latency.	software request from RUN0…3	system reset assertion, SAFE on hardware failure, RUN0…3 on interrupt event
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0…3	system reset assertion, SAFE on hardware failure, RUN0…3 on interrupt event or wakeup event

59.2 External Signal Description

The MC_ME has no connections to any external pins.

59.3 Memory Map and Register Definition

The MC_ME contains registers for:

- mode selection and status reporting

- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of additional core enabling and disabling control and status reporting

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

MC_ME memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Status Register (MC_ME_GS)	32	R	0C13_0000h	59.3.1/2295
4	Mode Control Register (MC_ME_MCTL)	32	R/W	3000_A50Fh	59.3.2/2297
8	Mode Enable Register (MC_ME_ME)	32	R/W	0000_801Dh	59.3.3/2299
C	Interrupt Status Register (MC_ME_IS)	32	w1c	0000_0000h	59.3.4/2301
10	Interrupt Mask Register (MC_ME_IM)	32	R/W	0000_0000h	59.3.5/2302
14	Invalid Mode Transition Status Register (MC_ME_IMTS)	32	w1c	0000_0000h	59.3.6/2304
18	Debug Mode Transition Status Register (MC_ME_DMTS)	32	R	0000_0000h	59.3.7/2306
20	RESET Mode Configuration Register (MC_ME_RESET_MC)	32	R	0013_0010h	59.3.8/2310
24	TEST Mode Configuration Register (MC_ME_TEST_MC)	32	R/W	0013_0010h	59.3.9/2313
28	SAFE Mode Configuration Register (MC_ME_SAFE_MC)	32	R/W	0093_0010h	59.3.10/ 2315
2C	DRUN Mode Configuration Register (MC_ME_DRUN_MC)	32	R/W	0013_0010h	59.3.11/ 2317
30	RUN0 Mode Configuration Register (MC_ME_RUN0_MC)	32	R/W	0013_0010h	59.3.12/ 2320
34	RUN1 Mode Configuration Register (MC_ME_RUN1_MC)	32	R/W	0013_0010h	59.3.13/ 2322
38	RUN2 Mode Configuration Register (MC_ME_RUN2_MC)	32	R/W	0013_0010h	59.3.14/ 2324
3C	RUN3 Mode Configuration Register (MC_ME_RUN3_MC)	32	R/W	0013_0010h	59.3.15/ 2327
40	HALT0 Mode Configuration Register (MC_ME_HALT0_MC)	32	R/W	0012_0010h	59.3.16/ 2329
48	STOP0 Mode Configuration Register (MC_ME_STOP0_MC)	32	R/W	0011_0010h	59.3.17/ 2331
60	Peripheral Status Register 0 (MC_ME_PS0)	32	R	0000_0000h	59.3.18/ 2334

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
64	Peripheral Status Register 1 (MC_ME_PS1)	32	R	0000_0000h	59.3.19/ 2336
68	Peripheral Status Register 2 (MC_ME_PS2)	32	R	0000_0000h	59.3.20/ 2338
6C	Peripheral Status Register 3 (MC_ME_PS3)	32	R	0000_0000h	59.3.21/ 2340
70	Peripheral Status Register 4 (MC_ME_PS4)	32	R	0000_0000h	59.3.22/ 2342
78	Peripheral Status Register 6 (MC_ME_PS6)	32	R	0000_0000h	59.3.23/ 2344
7C	Peripheral Status Register 7 (MC_ME_PS7)	32	R	0000_0000h	59.3.24/ 2346
80	Run Peripheral Configuration Register (MC_ME_RUN_PC0)	32	R/W	0000_0000h	59.3.25/ 2348
84	Run Peripheral Configuration Register (MC_ME_RUN_PC1)	32	R/W	0000_0000h	59.3.25/ 2348
88	Run Peripheral Configuration Register (MC_ME_RUN_PC2)	32	R/W	0000_0000h	59.3.25/ 2348
8C	Run Peripheral Configuration Register (MC_ME_RUN_PC3)	32	R/W	0000_0000h	59.3.25/ 2348
90	Run Peripheral Configuration Register (MC_ME_RUN_PC4)	32	R/W	0000_0000h	59.3.25/ 2348
94	Run Peripheral Configuration Register (MC_ME_RUN_PC5)	32	R/W	0000_0000h	59.3.25/ 2348
98	Run Peripheral Configuration Register (MC_ME_RUN_PC6)	32	R/W	0000_0000h	59.3.25/ 2348
9C	Run Peripheral Configuration Register (MC_ME_RUN_PC7)	32	R/W	0000_0000h	59.3.25/ 2348
A0	Low-Power Peripheral Configuration Register (MC_ME_LP_PC0)	32	R/W	0000_0000h	59.3.26/ 2350
A4	Low-Power Peripheral Configuration Register (MC_ME_LP_PC1)	32	R/W	0000_0000h	59.3.26/ 2350
A8	Low-Power Peripheral Configuration Register (MC_ME_LP_PC2)	32	R/W	0000_0000h	59.3.26/ 2350
AC	Low-Power Peripheral Configuration Register (MC_ME_LP_PC3)	32	R/W	0000_0000h	59.3.26/ 2350
B0	Low-Power Peripheral Configuration Register (MC_ME_LP_PC4)	32	R/W	0000_0000h	59.3.26/ 2350
B4	Low-Power Peripheral Configuration Register (MC_ME_LP_PC5)	32	R/W	0000_0000h	59.3.26/ 2350
B8	Low-Power Peripheral Configuration Register (MC_ME_LP_PC6)	32	R/W	0000_0000h	59.3.26/ 2350
BC	Low-Power Peripheral Configuration Register (MC_ME_LP_PC7)	32	R/W	0000_0000h	59.3.26/ 2350

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C9	LFAST_0 Peripheral Control Register (MC_ME_PCTL9)	8	R/W	00h	59.3.27/ 2351
CB	SIPI_0 Peripheral Control Register (MC_ME_PCTL11)	8	R/W	00h	59.3.28/ 2352
CC	ENET_0 Peripheral Control Register (MC_ME_PCTL12)	8	R/W	00h	59.3.29/ 2353
DE	PIT_0 Peripheral Control Register (MC_ME_PCTL30)	8	R/W	00h	59.3.30/ 2354
E4	DMAMUX_0 Peripheral Control Register (MC_ME_PCTL36)	8	R/W	00h	59.3.31/ 2355
E6	CRC_0 Peripheral Control Register (MC_ME_PCTL38)	8	R/W	00h	59.3.32/ 2356
10D	CAN_2 Peripheral Control Register (MC_ME_PCTL77)	8	R/W	00h	59.3.33/ 2357
10E	CAN_1 Peripheral Control Register (MC_ME_PCTL78)	8	R/W	00h	59.3.34/ 2358
10F	CAN_0 Peripheral Control Register (MC_ME_PCTL79)	8	R/W	00h	59.3.35/ 2359
11B	LINFlex_1 Peripheral Control Register (MC_ME_PCTL91)	8	R/W	00h	59.3.36/ 2360
122	DSPI_1 Peripheral Control Register (MC_ME_PCTL98)	8	R/W	00h	59.3.37/ 2361
123	DSPI_0 Peripheral Control Register (MC_ME_PCTL99)	8	R/W	00h	59.3.38/ 2362
128	SENT_0 Peripheral Control Register (MC_ME_PCTL104)	8	R/W	00h	59.3.39/ 2363
12B	FLEXRAY Peripheral Control Register (MC_ME_PCTL107)	8	R/W	00h	59.3.40/ 2364
13C	ADC_3 Peripheral Control Register (MC_ME_PCTL124)	8	R/W	00h	59.3.41/ 2365
13E	ADC_1 Peripheral Control Register (MC_ME_PCTL126)	8	R/W	00h	59.3.42/ 2366
149	ETIMER_1 Peripheral Control Register (MC_ME_PCTL137)	8	R/W	00h	59.3.43/ 2367
14D	CTU_1 Peripheral Control Register (MC_ME_PCTL141)	8	R/W	00h	59.3.44/ 2368
150	PWM_1 Peripheral Control Register (MC_ME_PCTL144)	8	R/W	00h	59.3.45/ 2369
152	DMAMUX_1 Peripheral Control Register (MC_ME_PCTL146)	8	R/W	00h	59.3.46/ 2370
18C	LINFlex_0 Peripheral Control Register (MC_ME_PCTL204)	8	R/W	00h	59.3.47/ 2371
190	DSPI_3 Peripheral Control Register (MC_ME_PCTL208)	8	R/W	00h	59.3.48/ 2372

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
191	DSPI_2 Peripheral Control Register (MC_ME_PCTL209)	8	R/W	00h	59.3.49/ 2373
196	SENT_1 Peripheral Control Register (MC_ME_PCTL214)	8	R/W	00h	59.3.50/ 2374
1AB	ADC_2 Peripheral Control Register (MC_ME_PCTL235)	8	R/W	00h	59.3.51/ 2375
1AD	ADC_0 Peripheral Control Register (MC_ME_PCTL237)	8	R/W	00h	59.3.52/ 2376
1AF	SGEN_0 Peripheral Control Register (MC_ME_PCTL239)	8	R/W	00h	59.3.53/ 2377
1B5	ETIMER_2 Peripheral Control Register (MC_ME_PCTL245)	8	R/W	00h	59.3.54/ 2378
1B7	ETIMER_0 Peripheral Control Register (MC_ME_PCTL247)	8	R/W	00h	59.3.55/ 2379
1BB	CTU_0 Peripheral Control Register (MC_ME_PCTL251)	8	R/W	00h	59.3.56/ 2380
1BF	PWM_0 Peripheral Control Register (MC_ME_PCTL255)	8	R/W	00h	59.3.57/ 2381
1C0	Core Status Register (MC_ME_CS)	32	R	0000_0001h	59.3.58/ 2382
1C4	Core Control Register (MC_ME_CCTL0)	16	R	00FEh	59.3.59/ 2383
1E0	CORE0 Address Register (MC_ME_CADDR0)	32	R/W	See section	59.3.60/ 2384

59.3.1 Global Status Register (MC_ME_GS)

This register contains global mode status.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_CURRENT_MODE				S_MTRANS	1	0		S_PDO	0		S_MVR	0		S_FLA	
W																
Reset	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								S_PLL1	S_PLL0	S_XOSC	S_IRC	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_GS field descriptions

Field	Description
0–3 S_CURRENT_ MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0

Table continues on the next page...

MC_ME_GS field descriptions (continued)

Field	Description
	1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
4 S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 S_PDO	Output power-down status — This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and the pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 S_FLA	Flash availability status 00 Flash is not available 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode and available for use
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 S_PLL1	secondary PLL status 0 secondary PLL is not stable 1 secondary PLL is providing a stable clock
25 S_PLL0	primary PLL status 0 primary PLL is not stable 1 primary PLL is providing a stable clock
26 S_XOSC	4-40 MHz crystal oscillator status

Table continues on the next page...

MC_ME_GS field descriptions (continued)

Field	Description
	0 4-40 MHz crystal oscillator is not stable 1 4-40 MHz crystal oscillator is providing a stable clock
27 S_IRC	16 MHz internal RC oscillator status 0 16 MHz internal RC oscillator is not stable 1 16 MHz internal RC oscillator is providing a stable clock
28–31 S_SYSCLK	System clock switch status — These bits specify the system clock currently used by the system. 0000 16 MHz IRCOSC 0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

59.3.2 Mode Control Register (MC_ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<mode>_MC registers must respect this for successful mode requests.

NOTE

Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TARGET_MODE				0																											
W																	KEY															
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

MC_ME_MCTL field descriptions

Field	Description
0–3 TARGET_MODE	Target chip mode — These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0…3 mode value. 0000 RESET (triggers a ‘functional’ reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 RESET (triggers a ‘destructive’ reset event)
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 KEY	Control key — These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101101011110000 (0x5AF0) INVERTED KEY: 1010010100001111 (0xA50F)

59.3.3 Mode Enable Register (MC_ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. RESET_FUNC, SAFE, DRUN, RUN0, and RESET_DEST modes are always enabled.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESET_DEST	0				STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET_FUNC
W	▨	▨					▨					▨	▨	▨		▨
Reset	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1

MC_ME_ME field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_ME field descriptions (continued)

Field	Description
16 RESET_DEST	‘destructive’ RESET mode enable 1 ‘destructive’ RESET mode is enabled
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HALT0	HALT0 mode enable 0 HALT0 mode is disabled 1 HALT0 mode is enabled
24 RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
25 RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
26 RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
27 RUN0	RUN0 mode enable 1 RUN0 mode is enabled
28 DRUN	DRUN mode enable 1 DRUN mode is enabled
29 SAFE	SAFE mode enable 1 SAFE mode is enabled
30 TEST	TEST mode enable 0 TEST mode is disabled 1 TEST mode is enabled
31 RESET_FUNC	‘functional’ RESET mode enable 1 ‘functional’ RESET mode is enabled

59.3.4 Interrupt Status Register (MC_ME_IS)

This register provides the current interrupt status.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										I_ICONF_CC	I_ICONF_CU	I_ICONF	I_IMODE	I_SAFE	I_MTC
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_IS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) — This bit is set if a write access to one of the ME_CADDRn registers is attempted while a mode transition is in progress. 0 No write to an ME_CADDRn register was attempted during an ongoing mode transition 1 A write to an ME_CADDRn register was attempted during an ongoing mode transition
27 I_ICONF_CU	Invalid mode configuration interrupt (Clock Usage) — This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration (clock usage) interrupt occurred 1 Invalid mode configuration (clock usage) interrupt is pending

Table continues on the next page...

MC_ME_IS field descriptions (continued)

Field	Description
28 I_ICONF	Invalid mode configuration interrupt — This bit is set whenever a write operation to ME_<mode>_MC registers with invalid mode configuration is attempted. It is cleared by writing a ‘1’ to this bit. 0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending
29 I_IMODE	Invalid mode interrupt — This bit is set whenever an invalid mode transition is requested. It is cleared by writing a ‘1’ to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
30 I_SAFE	SAFE mode interrupt — This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a ‘1’ to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
31 I_MTC	Mode transition complete interrupt — This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a ‘1’ to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, or STOP0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

59.3.5 Interrupt Mask Register (MC_ME_IM)

This register controls whether an event generates an interrupt or not.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										M_ICONF_CC	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_IM field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode configuration (core configuration) interrupt is masked 1 Invalid mode configuration (core configuration) interrupt is enabled
27 M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode configuration (core configuration) interrupt is masked 1 Invalid mode configuration (core configuration) interrupt is enabled
28 M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode configuration (core configuration) interrupt is masked 1 Invalid mode configuration (core configuration) interrupt is enabled
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
30 M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
31 M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

59.3.6 Invalid Mode Transition Status Register (MC_ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										S_MRIG	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_IMTS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_MRIG	Mode Request Ignored status — This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress. This bit is also set if a transition to STOP/HALT mode is requested when a wakeup is active. This bit is cleared by writing a ‘1’ to it. 0 Mode transition requested is not ignored 1 Mode transition requested is ignored
27 S_MTI	Mode Transition Illegal status — This bit is set whenever a new mode is requested while some other non-SAFE mode transition process is active (S_MTRANS is ‘1’). Please refer to Mode Transition Interrupts for the exceptions to this behavior. It is cleared by writing a ‘1’ to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
28 S_MRI	Mode Request Illegal status — This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a ‘1’ to this bit.

Table continues on the next page...

MC_ME_IMTS field descriptions (continued)

Field	Description
	0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
29 S_DMA	Disabled Mode Access status — This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
30 S_NMA	Non-existing Mode Access status — This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
31 S_SEA	SAFE Event Active status — This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a ‘1’ to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

59.3.7 Debug Mode Transition Status Register (MC_ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS.S_MTRANS may be taking longer than expected. NOTE The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0				MPH_BUSY	0		PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDP_PRPH_0_255	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	0	FLASH_SC	CDP_PRPH_224_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_DMTS field descriptions

Field	Description
0–3 PREVIOUS_ MODE	Previous chip mode — These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
8 MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator — This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its power-up/down sequencing. It is cleared when the MC_PCU has power-up/down sequencing. 0 Handshake is not busy 1 Handshake is busy
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PMC_PROG	MC_PCU Mode Change in Progress indicator — This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
12 DBG_MODE	Debug mode indicator — This bit is set while the chip is in debug mode. 0 The chip is not in debug mode 1 The chip is in debug mode
13 CCKL_PROG	Core Clock Enable/Disable in Progress — This bit is set while any core’s clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled 1 A core clock is being enabled or disabled
14 PCS_PROG	Progressive System Clock Switching in Progress This bit is set under any of these conditions: • PCS is in progress • Peripheral clock switching is ongoing • Core clock switching is ongoing • Power sequence process is ongoing (provided the target mode is not STANDBY) 0 PCS is not in progress 1 PCS is in progress
15 SMR	SAFE mode request from MC_RGM is active indicator — This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared. 0 A SAFE mode request is not active 1 A SAFE mode request is active
16 CDP_PRPH_0_255	Clock Disable Process Pending status for Peripherals 0 to 255 This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled.
17 VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
18 CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change.

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
	0 No state change is taking place 1 A state change is taking place
19 IRC_SC	IRC State Change during mode transition indicator — This bit is set when the 16 MHz internal RC oscillator is requested to change its power up/down state. It is cleared when the 16 MHz internal RC oscillator has completed its state change. 0 No state change is taking place 1 A state change is taking place
20 SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator — This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A ’secondary clock source’ is a clock source other than IRC.) 0 No state change is taking place 1 A state change is taking place
21 SYSCLK_SW	System Clock Switching pending status — 0 No system clock source switching is pending 1 A system clock source switching is pending
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FLASH_SC	FLASH State Change during mode transition indicator — This bit is set when the FLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
24 CDP_PRPH_ 224_255	Clock Disable Process Pending status for Peripherals 224…255 — This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. Peripheral n corresponds to the ME_PCTLn register. Please refer to the memory map for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
25 CDP_PRPH_ 192_223	Clock Disable Process Pending status for Peripherals 192…223 — This bit is set when any peripheral appearing in ME_PS6 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
26 CDP_PRPH_ 160_191	Clock Disable Process Pending status for Peripherals 160…191 — This bit is set when any peripheral appearing in ME_PS5 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
27 CDP_PRPH_ 128_159	Clock Disable Process Pending status for Peripherals 128…159 — This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
28 CDP_PRPH_96_ 127	Clock Disable Process Pending status for Peripherals 96…127 — This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
29 CDP_PRPH_64_ 95	Clock Disable Process Pending status for Peripherals 64…95 — This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
30 CDP_PRPH_32_ 63	Clock Disable Process Pending status for Peripherals 32…63 — This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
31 CDP_PRPH_0_ 31	Clock Disable Process Pending status for Peripherals 0…31 — This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

59.3.8 RESET Mode Configuration Register (MC_ME_RESET_MC)

This register configures system behavior during RESET mode.

NOTE

The following configuration values are set according to the chip configuration: XOSCON

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_RESET_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC 0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
	1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.9 TEST Mode Configuration Register (MC_ME_TEST_MC)

This register configures system behavior during TEST mode.

NOTE

Byte write accesses are not allowed to this register.

WARNING

Although the IRCON bit is writable, writing a ‘0’ to this bit will not turn off the 16 MHz internal RC oscillator is always in use. Writing a ‘0’ to this bit will cause the mode transition to not complete, since it will wait forever for the 16 MHz internal RC oscillator to become free.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W	▨				▨	▨	▨	▨		▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_TEST_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC

Table continues on the next page...

MC_ME_TEST_MC field descriptions (continued)

Field	Description
	0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.10 SAFE Mode Configuration Register (MC_ME_SAFE_MC)

This register configures system behavior during SAFE mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_SAFE_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control

Table continues on the next page...

MC_ME_SAFE_MC field descriptions (continued)

Field	Description
	0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC 0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.11 DRUN Mode Configuration Register (MC_ME_DRUN_MC)

This register configures system behavior during DRUN mode.

NOTE

Byte write accesses are not allowed to this register.

NOTE

The following configuration values are set according to the chip configuration: XOSCON

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_DRUN_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_DRUN_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC 0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

MC_ME_DRUN_MC field descriptions (continued)

Field	Description
	1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.12 RUN0 Mode Configuration Register (MC_ME_RUN0_MC)

This register configures system behavior during RUN0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_RUN0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC

Table continues on the next page...

MC_ME_RUN0_MC field descriptions (continued)

Field	Description
	0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.13 RUN1 Mode Configuration Register (MC_ME_RUN1_MC)

This register configures system behavior during RUN1 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_RUN1_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control

Table continues on the next page...

MC_ME_RUN1_MC field descriptions (continued)

Field	Description
	0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC 0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.14 RUN2 Mode Configuration Register (MC_ME_RUN2_MC)

This register configures system behavior during RUN1 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_RUN2_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RUN2_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC 0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

MC_ME_RUN2_MC field descriptions (continued)

Field	Description
	1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.15 RUN3 Mode Configuration Register (MC_ME_RUN3_MC)

This register configures system behavior during RUN3 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_RUN3_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC

Table continues on the next page...

MC_ME_RUN3_MC field descriptions (continued)

Field	Description
	0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.16 HALT0 Mode Configuration Register (MC_ME_HALT0_MC)

This register configures system behavior during HALT0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_HALT0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control

Table continues on the next page...

MC_ME_HALT0_MC field descriptions (continued)

Field	Description
	0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC 0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.17 STOP0 Mode Configuration Register (MC_ME_STOP0_MC)

This register configures system behavior during STOP0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_STOP0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_STOP0_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	primary PLL control 0 primary PLL is switched off 1 primary PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz IRCOSC 0001 4-40 MHz XOSC 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

MC_ME_STOP0_MC field descriptions (continued)

Field	Description
	1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

59.3.18 Peripheral Status Register 0 (MC_ME_PS0)

This register provides the status of the peripherals.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_PIT_0	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			S_ENET_0	S_SIPI_0	0	S_LFAST_0	0								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS0 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 S_PIT_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
2–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 S_ENET_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
20 S_SIPI_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 S_LFAST_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
23–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.3.19 Peripheral Status Register 1 (MC_ME_PS1)

This register provides the status of the peripherals.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									S_CRC_0	0	S_DMAMUX_0	0	S_ATX	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS1 field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 S_CRC_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME.

Table continues on the next page...

MC_ME_PS1 field descriptions (continued)

Field	Description
	0: Peripheral is frozen 1: Peripheral is active
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 S_DMAMUX_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 S_ATX	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.3.20 Peripheral Status Register 2 (MC_ME_PS2)

This register provides the status of the peripherals.

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				S_LINFlex_1	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_CAN_0	S_CAN_1	S_CAN_2	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS2 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_PS2 field descriptions (continued)

Field	Description
4 S_LINFlex_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
5–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 S_CAN_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
17 S_CAN_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
18 S_CAN_2	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
19–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.3.21 Peripheral Status Register 3 (MC_ME_PS3)

This register provides the status of the peripherals.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_ADC_1	0	S_ADC_3	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				S_FLEXRAY	0		S_SENT_0	0				S_DSPI_0	S_DSPI_1	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS3 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_PS3 field descriptions (continued)

Field	Description
1 S_ADC_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 S_ADC_3	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
4–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 S_FLEXRAY	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 S_SENT_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 S_DSPI_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
29 S_DSPI_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.3.22 Peripheral Status Register 4 (MC_ME_PS4)

This register provides the status of the peripherals.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													S_DMAMUX_1	0	S_PWM_1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		S_CTU_1	0			S_ETIMER_1	0								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS4 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 S_DMAMUX_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 S_PWM_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 S_CTU_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
19–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 S_ETIMER_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
23–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.3.23 Peripheral Status Register 6 (MC_ME_PS6)

This register provides the status of the peripherals.

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0									S_SENT_1	0				S_DSPI_2	S_DSPI_3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			S_LINFlex_0	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS6 field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_PS6 field descriptions (continued)

Field	Description
9 S_SENT_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
10–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 S_DSPI_2	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
15 S_DSPI_3	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 S_LINFlex_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
20–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.3.24 Peripheral Status Register 7 (MC_ME_PS7)

This register provides the status of the peripherals.

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_PWM_0	0			S_CTU_0	0			S_ETIMER_0	0	S_ETIMER_2	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_SGEN_0	0	S_ADC_0	0	S_ADC_2	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_PS7 field descriptions

Field	Description
0 S_PWM_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen

Table continues on the next page...

MC_ME_PS7 field descriptions (continued)

Field	Description
	1: Peripheral is active
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 S_CTU_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
5–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 S_ETIMER_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 S_ETIMER_2	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
11–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 S_SGEN_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 S_ADC_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 S_ADC_2	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
21–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.3.25 Run Peripheral Configuration Register (MC_ME_RUN_PC*n*)

The ME_RUN_PC0..7 registers configure eight different types of peripheral behavior during run modes.

Address: 0h base + 80h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_RUN_PC*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
25 RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active

Table continues on the next page...

MC_ME_RUN_PC*n* field descriptions (continued)

Field	Description
26 RUN1	Peripheral control during RUN1 0 Peripheral is frozen with clock gated 1 Peripheral is active
27 RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
28 DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
29 SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
30 TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated 1 Peripheral is active
31 RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

59.3.26 Low-Power Peripheral Configuration Register (MC_ME_LP_PC*n*)

The MC_LP_PC registers configure eight different types of peripheral behavior during non-run modes.

Address: 0h base + A0h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					STOP0	0	HALT0	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_LP_PC*n* field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HALT0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated 1 Peripheral is active
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

59.3.27 LFAST_0 Peripheral Control Register (MC_ME_PCTL9)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + C9h offset = C9h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL9 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.28 SIPI_0 Peripheral Control Register (MC_ME_PCTL11)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + CBh offset = CBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL11 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.29 ENET_0 Peripheral Control Register (MC_ME_PCTL12)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL12 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.30 PIT_0 Peripheral Control Register (MC_ME_PCTL30)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + DEh offset = DEh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL30 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.31 DMAMUX_0 Peripheral Control Register (MC_ME_PCTL36)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL36 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.32 CRC_0 Peripheral Control Register (MC_ME_PCTL38)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + E6h offset = E6h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL38 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.33 CAN_2 Peripheral Control Register (MC_ME_PCTL77)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 10Dh offset = 10Dh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL77 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.34 CAN_1 Peripheral Control Register (MC_ME_PCTL78)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 10Eh offset = 10Eh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL78 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.35 CAN_0 Peripheral Control Register (MC_ME_PCTL79)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 10Fh offset = 10Fh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL79 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.36 LINFlex_1 Peripheral Control Register (MC_ME_PCTL91)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 11Bh offset = 11Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL91 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.37 DSPI_1 Peripheral Control Register (MC_ME_PCTL98)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 122h offset = 122h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL98 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.38 DSPI_0 Peripheral Control Register (MC_ME_PCTL99)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 123h offset = 123h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL99 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.39 SENT_0 Peripheral Control Register (MC_ME_PCTL104)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 128h offset = 128h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL104 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.40 FLEXRAY Peripheral Control Register (MC_ME_PCTL107)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 12Bh offset = 12Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL107 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.41 ADC_3 Peripheral Control Register (MC_ME_PCTL124)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 13Ch offset = 13Ch

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL124 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.42 ADC_1 Peripheral Control Register (MC_ME_PCTL126)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 13Eh offset = 13Eh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL126 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.43 ETIMER_1 Peripheral Control Register (MC_ME_PCTL137)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 149h offset = 149h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL137 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.44 CTU_1 Peripheral Control Register (MC_ME_PCTL141)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 14Dh offset = 14Dh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL141 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.45 PWM_1 Peripheral Control Register (MC_ME_PCTL144)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 150h offset = 150h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL144 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.46 DMAMUX_1 Peripheral Control Register (MC_ME_PCTL146)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 152h offset = 152h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL146 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.47 LINFlex_0 Peripheral Control Register (MC_ME_PCTL204)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 18Ch offset = 18Ch

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL204 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.48 DSPI_3 Peripheral Control Register (MC_ME_PCTL208)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 190h offset = 190h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL208 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.49 DSPI_2 Peripheral Control Register (MC_ME_PCTL209)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 191h offset = 191h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL209 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.50 SENT_1 Peripheral Control Register (MC_ME_PCTL214)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 196h offset = 196h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL214 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.51 ADC_2 Peripheral Control Register (MC_ME_PCTL235)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1ABh offset = 1ABh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL235 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.52 ADC_0 Peripheral Control Register (MC_ME_PCTL237)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1ADh offset = 1ADh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL237 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.53 SGEN_0 Peripheral Control Register (MC_ME_PCTL239)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1AFh offset = 1AFh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL239 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.54 ETIMER_2 Peripheral Control Register (MC_ME_PCTL245)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1B5h offset = 1B5h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL245 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.55 ETIMER_0 Peripheral Control Register (MC_ME_PCTL247)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1B7h offset = 1B7h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL247 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.56 CTU_0 Peripheral Control Register (MC_ME_PCTL251)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1BBh offset = 1BBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL251 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.57 PWM_0 Peripheral Control Register (MC_ME_PCTL255)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 1BFh offset = 1BFh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write	▨							
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL255 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

59.3.58 Core Status Register (MC_ME_CS)

This register provides the status of each core.

Address: 0h base + 1C0h offset = 1C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															S_CORE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MC_ME_CS field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 S_CORE0	Core 0 status — This bits specifies the current status of the main and checker cores which are controlled by the ME_CCTL0 register. 0 main and checker cores are disabled 1 main and checker cores are running

59.3.59 Core Control Register (MC_ME_CCTL0)

This register indicates whether the main and checker cores are disabled or running during run modes.

Address: 0h base + 1C4h offset = 1C4h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	1	1	1	1	1	1	1	0

MC_ME_CCTL0 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 — main and checker cores are always disabled during STOP0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0 — main and checker cores are always disabled during HALT0.
8 RUN3	Core control during RUN3 0 main and checker cores are disabled with clock gated 1 main and checker cores are running
9 RUN2	Core control during RUN2 0 main and checker cores are disabled with clock gated 1 main and checker cores are running
10 RUN1	Core control during RUN1 0 main and checker cores are disabled with clock gated 1 main and checker cores are running
11 RUN0	Core control during RUN0 0 main and checker cores are frozen with clock gated 1 main and checker cores are running
12 DRUN	Core control during DRUN

Table continues on the next page...

MC_ME_CCTL0 field descriptions (continued)

Field	Description
	0 main and checker cores are frozen with clock gated 1 main and checker cores are running
13 SAFE	Core control during SAFE 0 main and checker cores are frozen with clock gated 1 main and checker cores are running
14 TEST	Core control during TEST 0 main and checker cores are frozen with clock gated 1 main and checker cores are running
15 RESET	Core control during RESET — main and checker cores are always disabled during RESET.

59.3.60 CORE0 Address Register (MC_ME_CADDR0)

This registers gives the boot address for main and checker cores and a bit for controlling whether main and checker cores are to be reset on the next mode change that has main and checker cores configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (that is, while the S_MTRANS bit of the ME_GS register = ‘1’). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

NOTE

> The reset value of the ADDR field of the ME_CADDR0 is determined by the chip configuration and boot mode.

Address: 0h base + 1E0h offset = 1E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADDR															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR														0	Reserved
W															▨	
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0

* Notes:

- ADDR field: Determined by the chip configuration and boot mode

MC_ME_CADDR0 field descriptions

Field	Description
0–29 ADDR	Core Address — This field is used by main and checker cores as the boot address (32-bit word aligned) when main and checker cores next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read/write field is reserved, and must always be written with the value '0'. Writing a '1' will lead to unpredictable behavior.

59.4 Functional Description

59.4.1 Mode Transition Request

The transition from one mode to another mode is normally handled by software by accessing the mode control register ME_MCTL. But the in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<mode>_MC register.The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to Mode Transition Interrupts.

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as RUN0…3 → RUN0…3, DRUN → DRUN, SAFE → SAFE, and TEST → TEST are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set till the status in the ME_GS register matches the configuration programmed in the respective ME_<mode>_MC register.

Note

> Software should poll the S_MTRANS bit in the ME_GS register after requesting a transition to HALT0 or STOP0 modes.

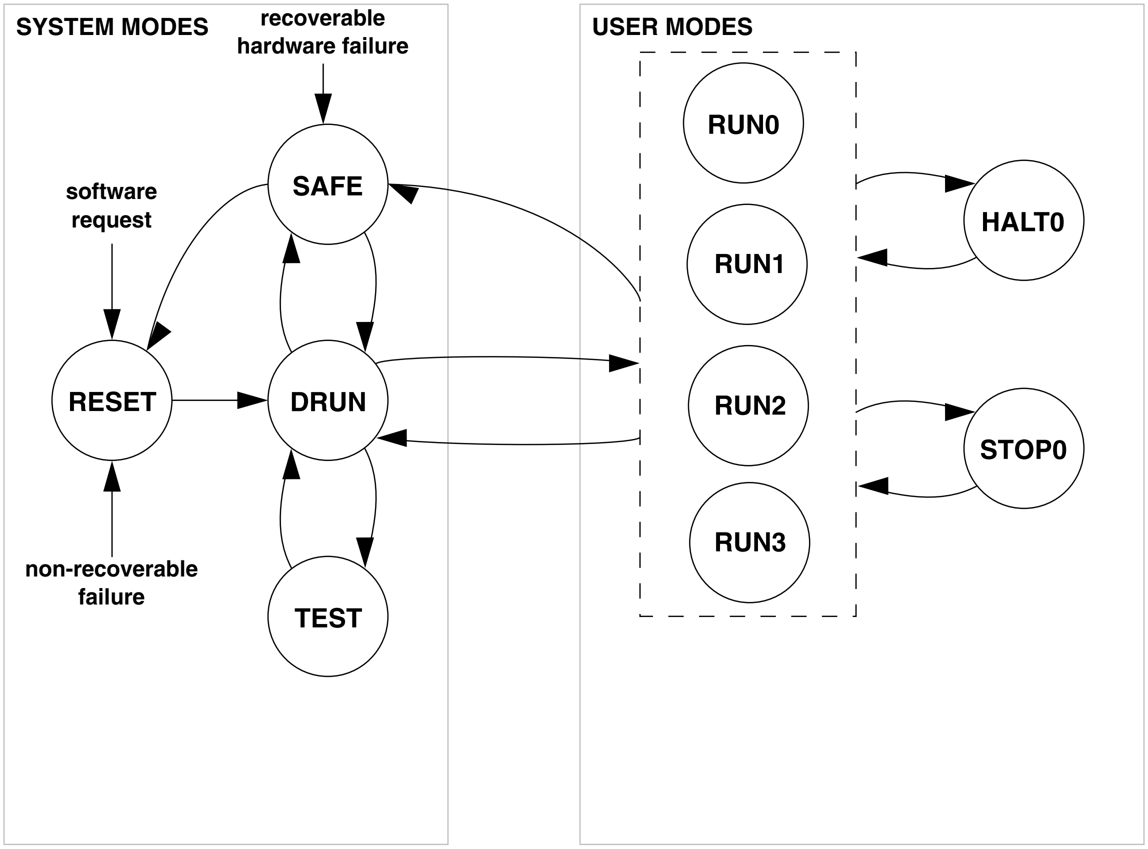


Figure 59-2. MC_ME Mode Diagram

59.4.2 Modes Details

59.4.2.1 RESET Mode

The chip enters this mode on the following events:

- from SAFE, DRUN, RUN0…3, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with either "0000" for a 'functional' reset or "1111" for a 'destructive' reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the ME_RESET_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

59.4.2.2 DRUN Mode

The chip enters this mode on the following events:

- automatically from RESET mode after completion of the reset sequence
- from RUN0…3, SAFE, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0011"

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz int. RC osc. as the system clock. The clock source configuration except system clock source 0 can be chosen by preloaded information in flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs

Note

> Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

59.4.2.3 SAFE Mode

The chip enters this mode on the following events:

- from any mode except RESET when the TARGET_MODE bit field of the ME_MCTL register is written with "0010"
- from any mode except RESET due to a SAFE mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see the MC_RGM chapter for details)

Note

> If a hardware SAFE mode request occurs during RESET, depending on the timing of the SAFE mode request, SAFE mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after DRUN entry. The SAFE mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a SAFE mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

If the SAFE mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the SAFE mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

Note

> If software requests to change to the SAFE mode and then requests to change back to the parent mode before the mode transition is completed, the chip's final mode after mode transition will be the SAFE mode, and an invalid mode transition interrupt will be generated.

As long as a SAFE event is active, the system remains in the SAFE mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either

- re-initialize the chip via the DRUN mode, or
- completely reset the chip via the RESET mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the PDO bit of the ME_SAFE_MC register should be set. The input levels remain unchanged.

59.4.2.4 TEST Mode

The chip enters this mode on the following events:

- from the DRUN mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0001"

As soon as any of the above events has occurred, a TEST mode transition request is generated. The mode configuration information for this mode is provided by the ME_TEST_MC register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the SYSCLK bit field to "1111", and in this case, the only way to exit this mode is via a chip reset.

This mode is intended to be used by software

- to execute software test routines

Note

Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

59.4.2.5 RUN0…3 Modes

The chip enters one of these modes on the following events:

- from the DRUN or another RUN0…3 mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0100…0111"
- from the HALT0 mode due to an interrupt event
- from the STOP0 mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a RUN0…3 mode transition request is generated. The mode configuration information for these modes is provided by the ME_RUN0…3_MC registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

Note

> Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

59.4.2.6 HALT0 Mode

The chip enters this mode on the following events:

- from one of the RUN0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with "1000".

As soon as any of the above events has occurred, a HALT0 mode transition request is generated. The mode configuration information for this mode is provided by ME_HALT0_MC register. This mode is quite configurable, and the ME_HALT0_MC register should be programmed according to the system needs. The flash can be put in low-power or power-down mode as needed. If there is a HALT0 mode request while an interrupt request is active, the transition to HALT0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

If a HALT0 mode transition request is generated at the same time that an interrupt request occurs there is a possibility that the system may hang requiring a power-on reset to recover. The application software should avoid the conditions where this system hang is possible. Before entering HALT0 mode the application software should disable all interrupts. If this is not easily manageable by software then the software should ensure that any transaction which can potentially generate an interrupt should be finished and all interrupts serviced before requesting HALT0 transition. As a result the wake up source from HALT0 mode should always be executed by an external interrupt and not from internal interrupts from the peripheral modules.

This mode is intended as a first-level low-power mode with

- the core clocks frozen the additional core clocks frozen
- XBAR masters cannot operate in system HALT0 mode as they are clock gated. Also, NPC should be enabled if mode transition is initiated while debugger is connected
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (i.e., within a few system clock cycles of an interrupt event)

Note

It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on HALT0 mode exit to ensure that the previous RUN0…3 mode configuration has been fully restored before executing critical code.

59.4.2.7 STOP0 Mode

The chip enters this mode on the following events:

- from one of the RUN0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with "1010".

As soon as any of the above events has occurred, a STOP0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP0_MC register. This mode is fully configurable, and the ME_STOP0_MC register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- the primary PLL
- the secondary PLL

The flash memory can be put in power-down mode as needed.

If there is a STOP0 mode request while any interrupt or wakeup event is active, the transition to STOP0 is aborted—with the resulting mode being the current mode—and an invalid mode transition interrupt is generated. The reason for this interrupt is S_MRIG (Mode request ignored): If a transition to SAFE (on SAFE mode request) is in progress, a transition request to STOP0 mode also generates an S_MRIG (Mode request ignored) interrupt. In case of a transition to DRUN (on reset), no interrupt is generated. Before attempting to enter STOP0 mode, properly disable all interrupts not intended to be used as wakeup events in software.

If a STOP0 mode transition request is generated at the same time that an interrupt request occurs there is a possibility that the system may hang requiring a power-on reset to recover. The application software should avoid the conditions where this system hang is possible. Before entering STOP0 mode the application software should disable all interrupts. If this is not easily manageable by software then the software should ensure that any transaction which can potentially generate an interrupt should be finished and all interrupts serviced before requesting STOP0 transition. As a result the wake up source from STOP0 mode should always be executed by an external wake-up/interrupt and not from internal interrupts from the peripheral modules.

This mode is intended as an advanced low-power mode with:

- the core clock frozen
- the additional core clocks frozen
- XBAR masters cannot operate in system STOP0 mode as they are clock gated. Also, NPC should be enabled if mode transition is initiated while debugger is connected
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started)

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the STOP0 mode, the 16 MHz internal RC oscillator clock is selected as the system clock until the target clock is available.

Note

> It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on STOP0 mode exit to ensure that the previous RUN0…3 mode configuration has been fully restored before executing critical code.

59.4.3 Mode Transition Process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

59.4.3.1 Target Mode Request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the TARGET_MODE bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to Mode Transition Interrupts for details.

In the case of mode transitions occurring because of hardware events such as a reset, a SAFE mode request, or interrupt requests and wakeup events to exit from low-power modes, the TARGET_MODE bit field of the ME_MCTL register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit S_MTRANS of the ME_GS register.

A RESET mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The RESET mode request has the highest priority, and the MC_ME is kept in the RESET mode during the entire reset sequence.

The SAFE mode request has the next highest priority after reset. It can be generated either by software via the ME_MCTL register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

59.4.3.2 Target Mode Configuration Loading

On completion of the Target Mode Request step, the target mode configuration from the ME_<target mode>_MC register is loaded to start the resources (voltage sources, clock sources, flash, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in the following table. A '√' indicates that a given resource is configurable for a given mode.

Table 59-2. MC_ME Resource Control Overview

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0…3	HALT0	STOP0
IRC	on	√ on	on	on	on	on	on
XOSC	off	√ off	off	√ off	√ off	√ off	√ off
PLL0	off	√ off	off	√ off	√ off	√ off	off
PLL1	off	√ off	off	√ off	√ off	√ off	√ off
FLASH	normal	√ normal	normal	√ normal	√ normal	√ low-power	√ power-down
MVREG	on	on	on	on	on	on	on
PDO	off	√ off	√ on	off	off	off	√ off

59.4.3.3 Peripheral Clocks Disable

On completion of the Target Mode Request step and the system clock frequency ramp-down of the System Clock Switching step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0…7 and ME_LP_PC0…7, and the peripheral control registers ME_PCTL***n***

Note

> The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a SAFE mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The SAFE mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to Peripheral Clock Gating for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the SAFE mode.

59.4.3.4 Processor Low-Power Mode Entry

If, on completion of the Peripheral Clocks Disable step, the mode transition is to the HALT0,STOP0 mode, the MC_ME requests each processor to enter its stopped state. Each processor acknowledges its stop state request bit after completing all outstanding bus transactions.

If, on completion of the Peripheral Clocks Disable step, the mode transition is from a non-low-power mode to another non-low-power mode, the MC_ME requests each processor which is configured in the ME_CCTL***n*** registers to be disabled to enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

59.4.3.5 Processor Clocks Disable

If, on completion of the Processor Low-Power Mode Entry step, and all processors which are configured in the ME_CCTL***n*** registers to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processors which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes.

Warning

> Clocks to the whole chip including the processor and system memory can be disabled in TEST mode.

59.4.3.6 System Clock Disable

If, on completion of the Processor Clocks Disable step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

59.4.3.7 Clock Sources Switch-On

On completion of the Processor Low-Power Mode Entry step, the MC_ME switches on all clock sources based on the <clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

- the 16 MHz internal RC oscillator
- the 4-40 MHz crystal oscillator
- the primary PLL
- the secondary PLL

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the S_<clock source> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

59.4.3.8 Flash Module Switch-On

On completion of the step, if the flash needs to be switched to normal mode from its low-power or power-down mode based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, the MC_ME requests the flash to exit from its low-power/power-down mode. When the flash is available for access, the S_FLA bit field of the ME_GS register is updated to "11" by hardware.

Warning

> It is illegal to switch the FLASH from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

59.4.3.9 Pad Outputs-On

On completion of the step, if the PDO bit of the ME_<target mode>_MC register is cleared, then

- all pad outputs are enabled to return to their previous state
- the I/O pads power sequence driver is switched on

59.4.3.10 Peripheral Clocks Enable

Based on the current and target chip modes, the peripheral configuration registers ME_RUN_PC0…7, ME_LP_PC0…7, and the peripheral control registers ME_PCTLn, the MC_ME enables the clocks for selected modules as required. This step is executed only after the process is completed, and if the value of the PWRLVL field of the ME_<target mode>_MC register is process is completed.

59.4.3.11 System and Processor Clocks Enable

On completion of the Flash Module Switch-On step, the MC_ME enables the non-processor system clock and the clocks of all processors that are configured in the ME_CCTL***n*** registers to be disabled in the current mode and to be running in the target mode.

If the value of the PWRLVL field of the ME_<target mode>_MC register is different from that of the ME_<current mode>_MC register, the mode change handshake will not be initiated until after the system clock frequency ramp-down step of the System Clock Switching process is completed.

59.4.3.12 Processor Low-Power Mode Exit

On completion of the System and Processor Clocks Enable step, the MC_ME requests all processors which are configured in the ME_CCTL*n* registers to be disabled in the current mode and to be running in the target mode.

59.4.3.13 System Clock Switching

Based on the SYSCLK bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz int. RC osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the 4-40 MHz crystal osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the primary PLL (PHI) takes effect only after the S_XOSC bit of the ME_GS register is set by hardware (i.e., the 4-40 MHz crystal oscillator has stabilized).
- The target clock configuration for the secondary PLL takes effect only after the S_PLL0 bit of the ME_GS register is set by hardware (i.e., the primary PLL has stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with "1111". This is possible only in theTEST mode.

If the value of the PWRLVL field of the ME_<target mode>_MC register not equal to that of the ME_<current mode>_MC register's, a progressive clock switching is performed in the MC_CGM to reduce the impact of a sudden increase or drop in the power consumption of the device. The MC_ME requests progressive clock switching and indicates the direction of the switching (ramp-up vs. ramp-down) via the clk_sys_sw_req and clk_sys_sw_ctl[2:0] outputs. If progressive clock switching has been requested, the system clock switching process is not completed until the MC_CGM has indicated that the progressive clock switching has finished by deasserting the clk_sys_sw_busy input ME_CLK_SYS_PCS.

The current system clock configuration can be observed by reading the S_SYSCLK bit field of the ME_GS register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the Clock Sources Switch-On process has completed if the target system clock source is one of the following:
  - the 16 MHz internal RC oscillator
  - the 4-40 MHz crystal oscillator
  - the primary PLL
  - the Secondary system clock source 1
- the Peripheral Clocks Disable process has completed in order not to change the system clock frequency before peripherals close their internal activitiesthe Peripheral Clocks Enable processe has completed

An overview of system clock source selection possibilities for each mode is shown in the following table. A '√' indicates that a given clock source is selectable for a given mode.

Table 59-3. MC_ME System Clock Selection Overview

System Clock Source	Mode						
	RESET	TEST	SAFE	DRUN	RUN0…3	HALT0	STOP0
16 MHz int. RC osc.	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)
4-40 MHz crystal osc.		√		√	√	√	√
primary PLL (PHI)		√		√	√	√	√
secondary PLL		√		√	√		
system clock is disabled		√[1]					

[1] disabling the system clock during TEST mode will require a reset in order to exit TEST mode

59.4.3.14 Pad Switch-Off

If the PDO bit of the ME_<target mode>_MC register is '1' then

- the outputs of the pads are forced to the high impedance state if the target mode is SAFE or TEST

This step is executed only after the Peripheral Clocks Disable process has completed.

59.4.3.15 Clock Sources (with no Dependencies) Switch-Off

Based on the chip mode and the <clock source>ON bits of the ME_<mode>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<clock source> of the ME_GS register to '0'. The following clock sources are switched off at this step:

- the 16 MHz internal RC oscillator, if it is not currently being used as the reference clock for PLL0 (the IRCOSC can be turned off only during TEST mode)
- the 4 MHz to 40 MHz crystal oscillator, if it is not currently being used as the reference clock for either PLL0 or PLL1
- the primary PLL, if it is not currently being used as the reference clock for PLL1
- the secondary PLL

For the clock sources related to the system clock, this step is executed only after the System Clock Switching process has completed.

59.4.3.16 Clock Sources (with Dependencies) Switch-Off

Based on the chip mode and the <clock source>ON bits of the ME_<mode>_MC registers, if a given clock source is to be switched off and all clock sources that need this clock source to be on have been switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<clock source> of the ME_GS register to '0'. The following clock sources are switched off at this step:

- the 16 MHz internal RC oscillator, if it is currently being used as the reference clock for PLL0 (the IRCOSC can be turned off only during TEST mode)
- the 4 MHz to 40 MHz crystal oscillator, if it is currently being used as the reference clock for either PLL0 or PLL1
- the primary PLL, if it is currently being used as the reference clock for PLL1

This step is executed only after

- the System Clock Switching process has completed in order not to lose the current system clock during mode transition
- the Clock Sources (with no Dependencies) Switch-Off process has completed in order to, for example, prevent unwanted clock transitions

59.4.3.17 Flash Switch-Off

Based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the flash is to be put in its low-power or power-down mode, the MC_ME requests the flash to enter the corresponding power mode and waits for the flash to acknowledge. The exact power mode status of the flash is updated in the S_FLA bit field of the ME_GS register. This step is executed only when the Processor Clocks Disable and System Clock Disable processes have completed.

59.4.3.18 Current Mode Update

The current mode status bit field S_CURRENT_MODE of the ME_GS register is updated with the target mode bit field TARGET_MODE of the ME_MCTL register when :

- all the updated status bits in the ME_GS register match the configuration specified in the ME_<target mode>_MC register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

Note

SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the ME_GS.S_RC bit to be set. This is to ensure that the SAFE mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the S_MTRANS bit of the ME_GS register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the ME_DMTS register can indicate which process is still in progress.

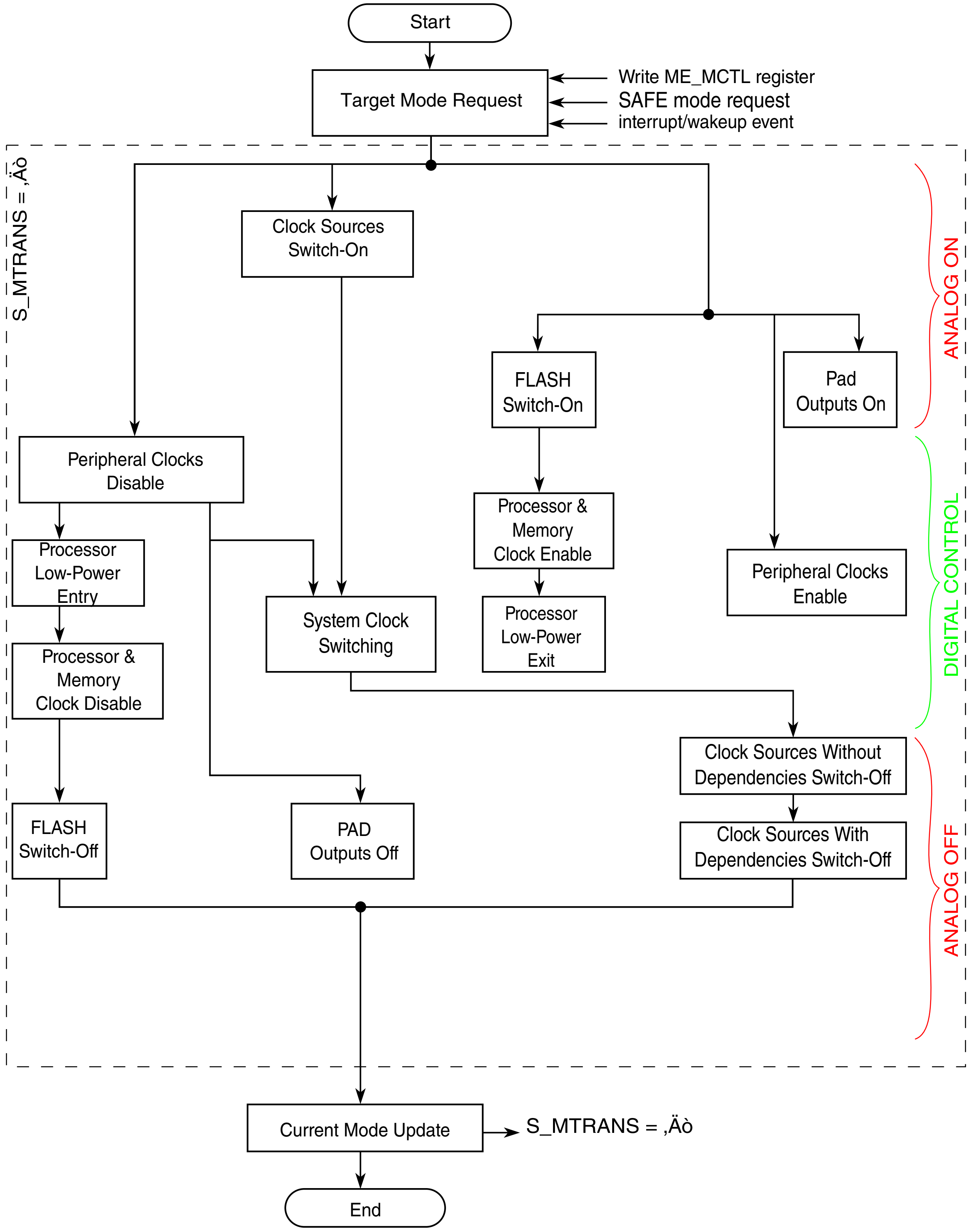


Figure 59-3. MC_ME Transition Diagram

59.4.4 Protection of Mode Configuration Registers

While programming the mode configuration registers ME_<mode>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the 16 MHz internal RC oscillator is selected as the system clock, IRC must be on.
- If the 4 MHz to 40 MHz crystal oscillator clock is selected as the system clock, XOSC must be on.
- If the primary PLL (PHI) clock is selected as the system clock, PLL0 must be on.
- If the secondary PLL clock is selected as the system clock, PLL1 must be on.
- If the 16 MHz internal RC oscillator is selected as the reference for PLL0 (via the MC_CGM_AC3_SC register in the MC_CGM) and PLL0 is on, IRCOSC must be on.
- If the 4 MHz to 40 MHz crystal oscillator is selected as the reference for PLL0 (via the MC_CGM_AC3_SC register in the MC_CGM) and PLL0 is on, XOSC must be on.
- If the 4 MHz to 40 MHz crystal oscillator is selected as the reference for PLL1 (via the MC_CGM_AC4_SC register in the MC_CGM) and PLL1 is on, XOSC must be on.
- If the primary PLL is selected as the reference for PLL1 (via the MC_CGM_AC4_SC register in the MC_CGM) and PLL1 is on, PLL0 must be on.

Note

Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

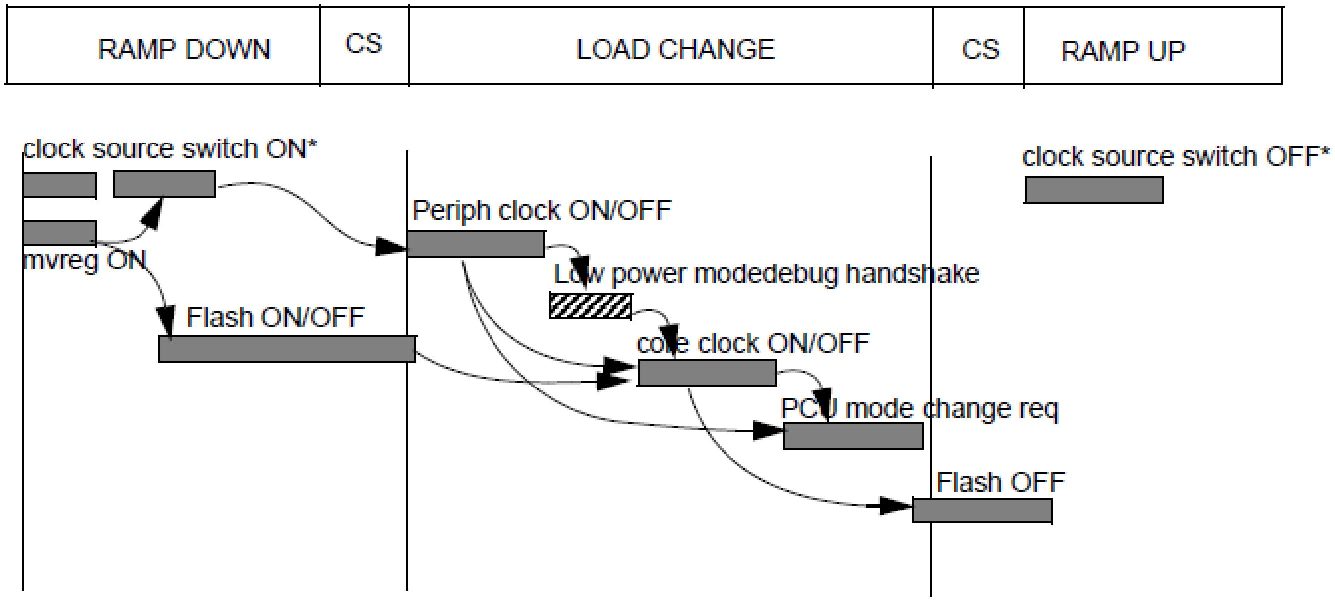
- Configuration "00" for the FLAON bit field is reserved.
- System clock configurations marked as 'reserved' may not be selected.
- Configuration "1111" for the SYSCLK bit field is allowed only for theTEST mode, and only in this case may all system clock sources be turned off.

Warning

> If the system clock is stopped during TEST mode, the chip can exit only via a system reset.

59.4.5 Mode transition and progressive clock switching

The MC_ME initiates a progressive clock switching via MC_CGM whenever the **ME_<current mode>_MC.PWRLVL** field is different from **ME_<target mode>_MC.PWRLVL** bit field. Broadly, the mode transition process can be divided into three steps with respect to progressive cclock switching - RAMP DOWN, LOAD CHANGE and RAMP UP. One or more of these steps may not be needed based on the current and target mode configirations. The clock switching (from current system clock source to target system clock source) can happen before or after the LOAD CHANGE step.



* - The clock source switching ON process is shown in two steps - the main vreg independent clock source switching ON and the main vreg dependent clock sources swicthing ON.

CS - Clock swtching. Only one clock switching step is needed.It can be before the load change process or after it.

▨ Optional step

Figure 59-4. MC_ME mode transition process with PCS

The table below shows the different ways the mode transition process can complete.

Table 59-4. Mode transition steps

Pwrlvl	Source clock	Target clock	Mode transition steps
Same	IRC	Non-IRC	LoadChange -> ClockSwitch
	IRC	IRC	LoadChange
	Non-IRC	IRC	LoadChange -> ClockSwitch
	Non-IRC	Non-IRC	LoadChange -> ClockSwitch
Different	IRC	Non-IRC	LoadChange -> ClockSwitch -> RampUp
	IRC	IRC	LoadChange
	Non-IRC	IRC	RampDown -> ClockSwitch -> LoadChange
	Non-IRC	Non-IRC	RampDown -> ClockSwitch -> LoadChange -> RampUp

59.4.6 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a SAFE mode transition not due to a software request, and indicating when a mode transition has completed.

59.4.6.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the ME_<mode>_MC registers violating the protection rules mentioned in the Protection of Mode Configuration Registers, the interrupt pending bit I_ICONF of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF of the ME_IM register is '1'.

In addition, during a mode transition, if a clock source has been configured in the ME_<target mode>_MC register to be off and a peripheral requiring this clock source to be on has been enabled via the ME_RUN_PC0…7/ME_LP_PC0…7 and ME_PCTLn registers, the interrupt pending bit I_ICONF_CU of the ME_IS register is set and an iterrupt request is generated if the mask bit M_ICONF_CU of the ME_IM register is '1'.

If an attempt to write to one of the ME_CADDRn registers is made after a mode change request has been made via the second write to the ME_CTL register and before the completion of that mode transition as indicated by the S_MTRANS bit in the ME_GS register deasserting, the interrupt pending bit I_ICONF_CC of the ME_IS register is set and an iterrupt request is generated if the mask bit M_ICONF_CC of the ME_IM register is '1'.

59.4.6.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the SAFE mode and the SAFE mode request from MC_RGM is active, and if the target mode requested is other than RESET or SAFE, then this new mode request is considered to be invalid, and the S_SEA bit of the ME_IMTS register is set.
- If the TARGET_MODE bit field of the ME_MCTL register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the S_NMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If some of the chip modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the S_DMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit S_MRI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition to a mode other than SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is '1'), the mode transition illegal status bit S_MTI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a new mode request occur while a mode transition to the SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is ‘1’), the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.

Note

As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA, S_MRI, S_MRIG, and S_MTI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the RESET or SAFE mode irrespective of the mode transition status.
- As the exit of HALT0 and STOP0 modes depends on the interrupts of the system which can occur at any instant, these requests to return to RUN0…3 modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software-determined 'reasonable' amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.

CAUTION

Depending on the cause of an incomplete mode transition, aborting by requesting the parent mode might not succeed. In this case, software must request a change to either SAFE or RESET mode, because the cause of the lockup is most likely a hardware fault.

- Self-transition requests (e.g., RUN0 → RUN0) are not considered as invalid even when the mode transition process is active (i.e., S_MTRANS is '1'). During the low-power mode exit process, if the system is not able to enter the respective RUN0…3 mode properly (i.e., all status bits of the ME_GS register match with configuration bits in the ME_<mode>_MC register), then software can only request the SAFE or RESET mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit I_IMODE of the ME_IS register is set, and an interrupt request is generated if the mask bit M_IMODE of the ME_IM register is '1'.

59.4.6.3 SAFE Mode Transition Interrupt

Whenever the system enters the SAFE mode as a result of a SAFE mode request from the MC_RGM due to a hardware failure, the interrupt pending bit I_SAFE of the ME_IS register is set, and an interrupt is generated if the mask bit M_SAFE of ME_IM register is '1'.

The SAFE mode interrupt pending bit can be cleared only when the SAFE mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a SAFE mode request). If the system is already in SAFE mode, any new SAFE mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the SAFE mode interrupt pending bit is not set when the SAFE mode is entered by a software request (i.e., programming of ME_MCTL register).

59.4.6.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (i.e., the S_MTRANS bit of ME_GS register transits from '1' to '0'), the interrupt pending bit I_MTC of the ME_IS register is set, and an interrupt request is generated if the mask bit M_MTC of the ME_IM register is '1'. The interrupt bit I_MTC is not set when entering low-power modes HALT0 and STOP0 in order to avoid the same event requesting the immediate exit of these low-power modes.

59.4.7 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0…7 are chosen only during the software running modes DRUN, TEST, SAFE, and RUN0…3. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTL***n*** registers.

The low-power peripheral configuration registers ME_LP_PC0…7 are chosen only during the low-power modes HALT0 and STOP0. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTL***n*** registers.

Any modifications to the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTL***n*** registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the chip enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTL***n*** register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits.

59.4.8 Application Example

The following figure shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

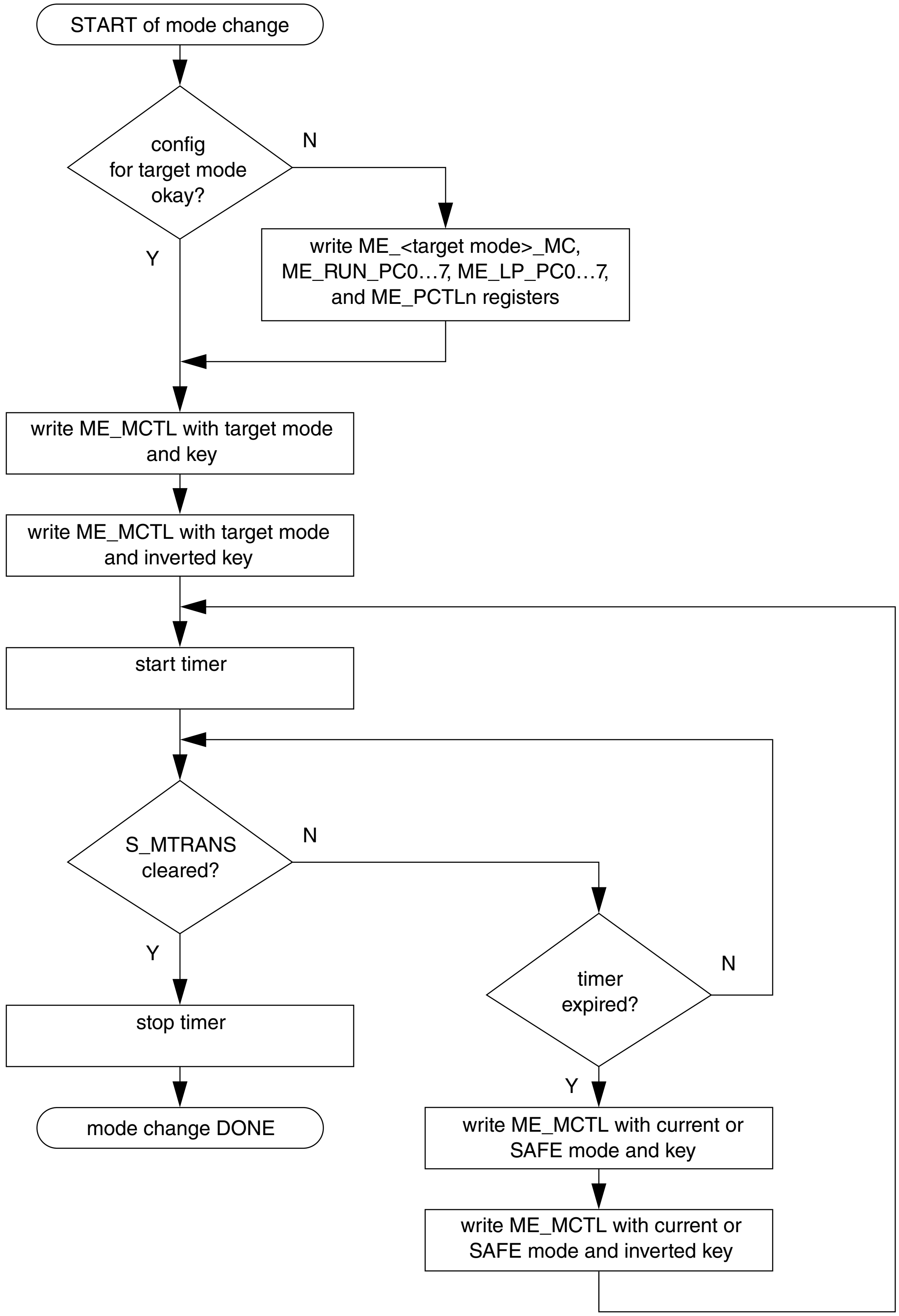

START of mode change
config
for target mode
okay?
N
Y
write ME_<target mode>_MC,
ME_RUN_PC0…7, ME_LP_PC0…7,
and ME_PCTLn registers
write ME_MCTL with target mode
and key
write ME_MCTL with target mode
and inverted key
start timer
S_MTRANS
cleared?
N
Y
timer
expired?
N
Y
stop timer
mode change DONE
write ME_MCTL with current or
SAFE mode and key
write ME_MCTL with current or
SAFE mode and inverted key

Figure 59-5. MC_ME Application Example Flow Diagram

Chapter 60
Core Debug Support

60.1 Overview

Internal debug support in the e200z4251n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *Power ISA 2.06*. e200z4251n3 supports a subset of these defined facilities. In addition to the facilities defined in *Power ISA 2.06*, e200z4251n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

60.1.1 Software Debug Facilities

e200z4251n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors which are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($\mathrm{DBCR0}_{\mathrm{IDM}}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by $\mathrm{MSR}_{\mathrm{DE}}$, these recorded exceptions cause Debug interrupts to occur. When $\mathrm{DBCR0}_{\mathrm{IDM}}$ is cleared (and $\mathrm{EDBCR0}_{\mathrm{EDM}}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $\mathrm{DBCR0}_{\mathrm{IDM}}$ is cleared (or is overridden by $\mathrm{EDBCR0}_{\mathrm{EDM}}$ being set and EDBRAC0 indicating no resource is "owned" by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

60.1.1.1 *Power ISA 2.06* Compatibility

The e200z4251n3 core implements a subset of the *Power ISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

60.1.2 Additional Debug Facilities

In addition to the debug functionality defined in *Power ISA 2.06*, e200z4251n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See Sharing Debug Resources by Software/Hardware.) A data trace port is also provided.

60.1.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z4251n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the "Data Trace Control Register (DTC)" section in the Core (e200z4251n3) Nexus 3 Module chapter. The data trace port will report aligned accesses, and most misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0. Also, for accesses generated by a LSP load/store instruction using circular addressing mode which crosses a buffer boundary, the two portions of the misaligned access will be reported separately, since otherwise it is not possible to correctly correlate the data with the initial access address.

60.1.3 Hardware Debug Facilities

The e200z4251n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard IEEE 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z4251n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the "hardware-owned" debug facilities. In addition, since resources are "owned" by the hardware debugger, inconsistent values may be present if software attempts to read "hardware-owned" debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in Debug registers are reserved for use by the external debugger. The same events described in Software Debug Events and Exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

60.1.4 Sharing Debug Resources by Software/Hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When EDBCR0$_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when DBCR0$_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR$_{DE}$. Hardware-owned resources which set EDBSR0 bits when EDBCR0$_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, (EDBCR0$_{EDM}$=1 and EDBRAC0$_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

60.1.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a "hardware-owned" resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

The following figure shows the e200z4251n3 debug resources.

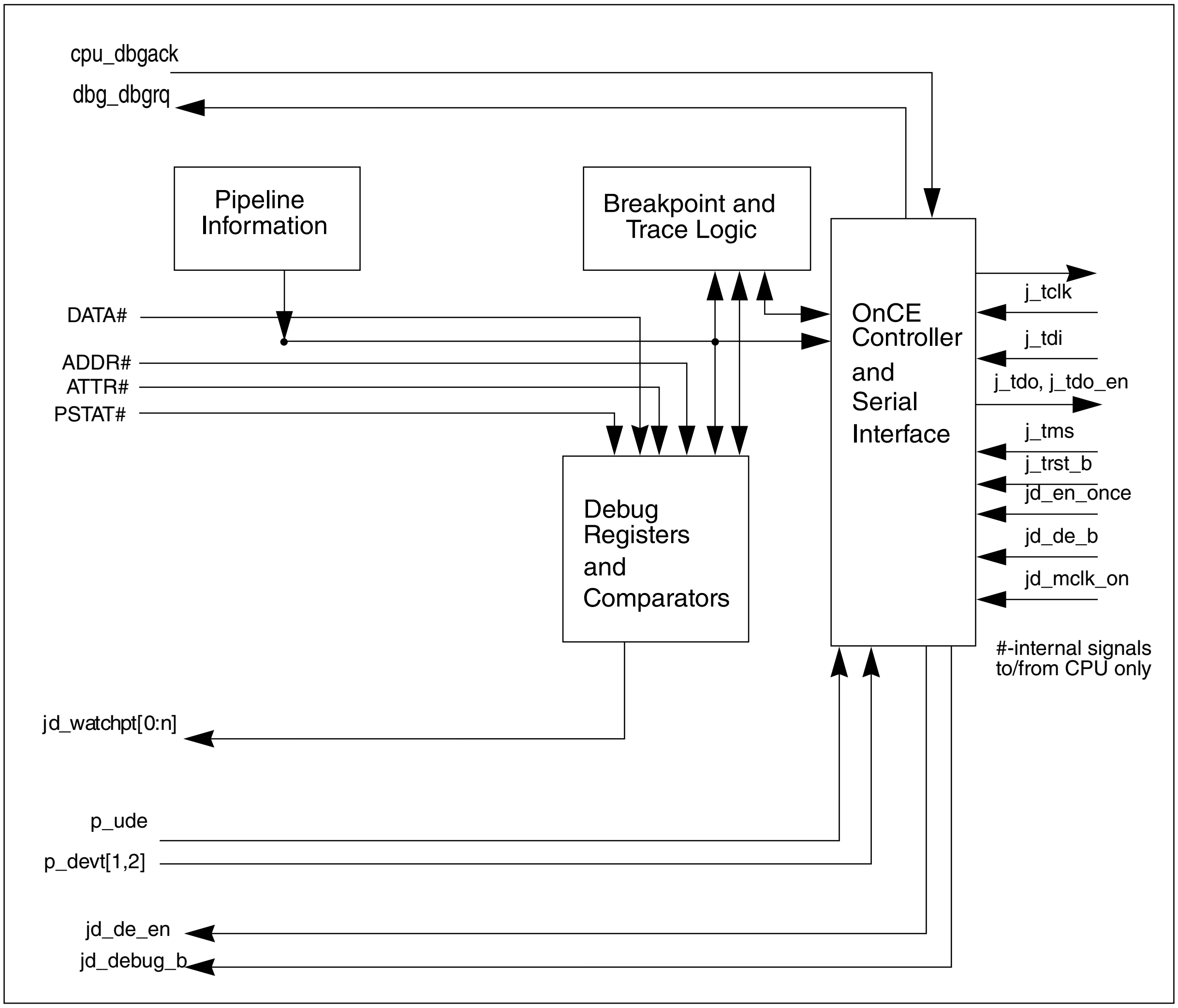


Figure 60-1. e200z4251n3 Debug Resources

60.2 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled (DBCR0$_{\text{IDM}}$=1) and not overridden by external debug mode (EDBCR0$_{\text{EDM}}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is "owned" by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR$_{\text{DE}}$, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *Power ISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z4251n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in External debug event.
- The Performance Monitor Interrupt event PMI which is described in Performance Monitor Interrupt debug event.

The e200z4251n3 debug configuration supports most of these event types. Unsupported *Power ISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

60.2.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

60.2.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note

> In contrast to the *Power ISA 2.06* definition, Data Address Compare events on e200z4251n3 do not prevent the load or store class instruction from completing. If a load or store class

instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting DBSR_{IDE}) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $\text{DBSR}_{\text{DAC\_OFST}}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the DBSR_{IDE} bit will be set. In addition to occurring when $\text{DBCR0}_{\text{IDM}}$=1, this circumstance can also occur when $\text{EDBCR0}_{\text{EDM}}$=1 and the event is hardware-owned, in which case $\text{EDBSR0}_{\text{IDE}}$ will be set.

Note

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note

DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, i.e. no data comparisons are performed since these instructions do not have associated data values.

Note

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note

DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.

Note

DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note

Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/ EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

60.2.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/ store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $\text{DBSR}_{\text{DAC\_OFST}}$ field when the DVC DAC status is recorded.

Table 60-1 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in Table 60-1, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 60-1. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Ldst instruction,no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table continues on the next page...

Table 60-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DVC DACy,Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy,Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction(no exception)	DSI, with or without DAC,Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction(no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 60-2 through Table 60-5 show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 60-2. DAC events and Resultant Updates, Dual-issue case 1

Instruction Sequence:	Event(s)	Result
The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu		
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)

	Instruction (1):
	DVC DACx
	Instruction (3):
	DSI, with or without DAC
		Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6).
		Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DACy
		Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DVC DACy
		Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DVC DACy
	Instruction (5):
	DSI, with or without DAC
		Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
		Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table continues on the next page...

Table 60-2. DAC events and Resultant Updates, Dual-issue case 1 (continued)

Instruction Sequence: **The following pairs dual-issue:** **(1) load/store** **(2) alu** **(3) load/store** **(4) alu** **(5) load/store** **(6) alu**	**Event(s)**	**Result**

	Instruction (1):
	DVC DACx
	Instruction (3):
	DVC DACy
	Instruction (5):
	DACy or DVC DACy
		Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 60-3. DAC events and Resultant Updates, Dual-issue case 2

Instruction Sequence: **The following pairs dual-issue:** **(1) load/store** **(2) alu** **(3) load/store** **(4) alu** **(5) alu** **(6) load/store**	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6).

Table continues on the next page...

Table 60-3. DAC events and Resultant Updates, Dual-issue case 2 (continued)

Instruction Sequence: **The following pairs dual-issue:** **(1) load/store** **(2) alu** **(3) load/store** **(4) alu** **(5) alu** **(6) load/store**	**Event(s)**	**Result**
	Instruction (3): DSI, with or without DAC	**Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DACy
		Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DVC DACy
		Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update.
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DVC DACyInstruction (6):
	DSI, with or without DAC
		Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
		Note: in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DVC DACy
	Instruction (6):
	DACy or DVC DACy
		Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 60-4. DAC events and Resultant Updates, Dual-issue case 3

Instruction Sequence: **The following pairs dual-issue:** **(1) load/store** **(2) alu** **(3) alu** **(4) alu** **(5) load/store** **(6) alu**	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).

	Instruction (1):
	DVC DACx
	Instruction (5):
	DSI, with or without DAC
		Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6).
		Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

	Instruction (1):
	DVC DACx
	Instruction (5):
	DACy
		Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

	Instruction (1):
	DVC DACx
	Instruction (5):
	DVC DACy
		Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 60-5. DAC events and Resultant Updates, Dual-issue case 4

Instruction Sequence: **The following pairs dual-issue:** **(1) load/store** **(2) alu** **(3) load/store** **(4) alu** **(5) alu** **(6) alu**	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)

	Instruction (1):
	DVC DACx
	Instruction (3):
	DSI, with or without DAC
		Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6).
		Note: in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DACy
		Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

	Instruction (1):
	DVC DACx
	Instruction (3):
	DVC DACy
		Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6).
		Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

60.2.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note

> Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

60.2.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

60.2.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the DBSR_{BRT} bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

60.2.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($\text{DBCR0}_{\text{ICMP}}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $\text{DBSR}_{\text{ICMP}}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus DBSR_{IDE} is not generally set by an ICMP debug event.

One circumstance may cause the $\text{DBSR}_{\text{ICMP}}$ and DBSR_{IDE} bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and DBSR_{IDE} is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $\text{DBCR0}_{\text{IDM}}$=1, this circumstance can also occur when $\text{EDBCR0}_{\text{EDM}}$=1 and the event is hardware-owned, in which case $\text{EDBSR0}_{\text{IDE}}$ will be set.

Note

> Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $\text{DBCR0}_{\text{ICMP}}$=1, nor by the execution of an instruction that sets $\text{DBCR0}_{\text{ICMP}}$ to '1' while MSR_{DE}=1.

60.2.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

60.2.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

60.2.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

60.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

60.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

60.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

60.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

60.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

60.3.1 Debug Address and Value Registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in the following figure) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in the following figure) corresponds to byte offset 7, F, … . When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z4251n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

<table>
<tr><td colspan="8">B0</td><td colspan="8">B1</td><td colspan="8">B2</td><td colspan="8">B3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="8">B4</td><td colspan="8">B5</td><td colspan="8">B6</td><td colspan="8">B7</td></tr>
<tr><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>47</td><td>48</td><td>49</td><td>50</td><td>51</td><td>52</td><td>53</td><td>54</td><td>55</td><td>56</td><td>57</td><td>58</td><td>59</td><td>60</td><td>61</td><td>62</td><td>63</td></tr>
</table>

SPR - 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset - Unaffected

Figure 60-2. DVC1, DVC2 registers

60.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z4251n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

60.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z4251n3 adds some implementation specific bits to this register, as seen in the following figure.

EDM	IDM	RST		ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1		DAC2		RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	0		CIRPT	CRET	0				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 308; Read/Write; Reset[1] - 0x0

Figure 60-3. Debug Control Register 0 (DBCR0) register

Note

[1] DBCR0$_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if DBCR0$_{EDM}$=0, as well as unconditionally by **m_por**. If DBCR0$_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the DBCR0$_{RST}$ field will be reset by **p_reset_b**. The DBCR0$_{RST}$ field will always be reset by **p_reset_b** regardless of the value of DBCR0$_{EDM}$.

The following provides bit definitions for Debug Control Register 0.

Table 60-6. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled

Table continues on the next page...

Table 60-6. DBCR0 field descriptions (continued)

Bit	Name	Description
		1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1- IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled

Table continues on the next page...

Table 60-6. DBCR0 field descriptions (continued)

Bit	Name	Description
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

60.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in the following figure.

IAC1US		IAC1ER		IAC2US		IAC2ER		IAC12M		0						IAC3US		IAC3ER		IAC4US		IAC4ER		IAC34M		0					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 309; Read/Write; Reset[1] - 0x0

Figure 60-4. DBCR1 Register

Note

[1] Reset by processor reset **p_reset_b** if $\text{EDBCR0}_{\text{EDM}}$=0, as well as unconditionally by **m_por**. If $\text{EDBCR0}_{\text{EDM}}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 1.

Table 60-7. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode

Table continues on the next page...

Table 60-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address

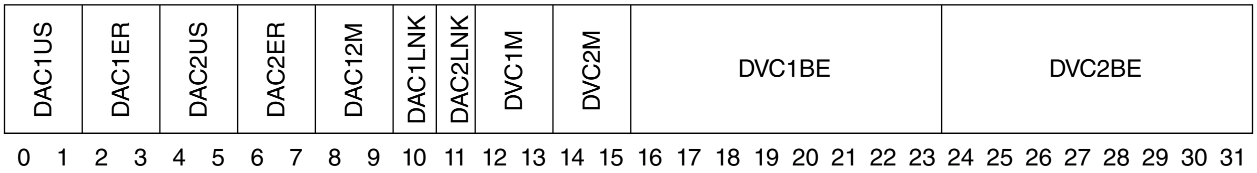
Table continues on the next page...

Table 60-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

60.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in the following figure.



SPR - 310; Read/Write; Reset[1] - 0x0

Figure 60-5. DBCR2 Register

Note

[1] Reset by processor reset **p_reset_b** if $\text{EDBCR0}_{\text{EDM}}$=0, as well as unconditionally by **m_por**. If $\text{EDBCR0}_{\text{EDM}}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 2.

Table 60-8. DBCR2 Bit Definitions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.

Table continues on the next page...

Table 60-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies Exact Address Compare since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR
12:13	DVC1M	Data Value Compare 1 Mode When DBCR4$_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When DBCR4$_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Note: Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When DBCR4$_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access.

Table continues on the next page...

Table 60-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When DBCR4$_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2.

Table 60-8. DBCR2 Bit Definitions

Bit	Name	Description
		xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

60.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in the following figure.

0	1	2	3	4 5 6 7 8 9 10 11	12	13	14	15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
0	DVC1C	0	DVC2C	0	0	DAC1XMH	0	DAC2XMH	DAC1XM	DAC2XM	DAC1CFG	DAC2CFG

SPR - 563; Read/Write; Reset[1] - 0x0

Figure 60-6. DBCR4 Register

Note

[1] DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 4.

Table 60-9. DBCR4 field description

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation.

Table continues on the next page...

Table 60-9. DBCR4 field description (continued)

Bit	Name	Description
		1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field. 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. . DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $\text{DBCR0}_{\text{DAC1}}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $\text{DBCR0}_{\text{DAC1}}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $\text{DBCR0}_{\text{DAC1}}$=00 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $\text{DBCR0}_{\text{DAC1}}$ and $\text{DBCR2}_{\text{DVC1M}}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint).

Table continues on the next page...

Table 60-9. DBCR4 field description (continued)

Bit	Name	Description
		NOTE: unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **NOTE:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

60.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in the following figure.

.

IAC5US		IAC5ER		IAC6US		IAC6ER		IAC56M		0						IAC7US		IAC7ER		IAC8US		IAC8ER		IAC78M		0					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 564; Read/Write; Reset[1] - 0x0

Figure 60-7. DBCR5 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 5.

Table 60-10. DBCR5 field descriptions

Bit(s)	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved

Table continues on the next page...

Table 60-10. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
		10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

60.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in the following figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
IAC1XM				IAC2XM				IAC3XM				IAC4XM				IAC5XM				IAC6XM				IAC7XM				IAC8XM			

SPR - 603; Read/Write; Reset[1] - 0x0

Figure 60-8. DBCR6 Register

Note

[1] DBCR6 is reset by processor reset **p_reset_b** if $\text{EDBCR0}_{\text{EDM}}$=0, as well as unconditionally by **m_por**. If $\text{EDBCR0}_{\text{EDM}}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 6.

Table 60-11. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Controll. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported.

Table continues on the next page...

Table 60-11. DBCR6 field descriptions (continued)

Bit	Name	Description
		1101 - 1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

60.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in the following figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0												0	DAC3XMH	0	DAC4XMH	DAC3XM				DAC4XM				DAC3CFG				DAC4CFG			

SPR - 596; Read/Write; Reset[1] - 0x0

Figure 60-9. DBCR7 Register

Note

[1] DBCR7 is reset by processor reset **p_reset_b** if EDBCR0$_{EDM}$=0, as well as unconditionally by **m_por**. If EDBCR0$_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 7.

Table 60-12. DBCR7 Field Descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High .DAC3XMH extends the range of the DAC3XM field 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. . DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses

Table continues on the next page...

Table 60-12. DBCR7 Field Descriptions (continued)

Bit	Name	Description
		0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001 - 1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

60.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in the following figure.

DAC3US	DAC3ER	DAC4US	DAC4ER	DAC34M	DAC3LNK	DAC4LNK	0
0 1	2 3	4 5	6 7	8 9	10	11	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 597; Read/Write; Reset[1] - 0x0

Figure 60-10. DBCR8 Register

Note

[1] Reset by processor reset **p_reset_b** if EDBCR0$_{EDM}$=0, as well as unconditionally by **m_por**. If EDBCR0$_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 8.

Table 60-13. DBCR8 Bit Definitions

Bit(s)	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 - DAC3 debug events not affected by MSR$_{PR}$ 01 - Reserved 10 - DAC3 debug events can only occur if MSR$_{PR}$=0 (Supervisor mode) 11 - DAC3 debug events can only occur if MSR$_{PR}$=1. (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 - DAC3 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC3 debug events are based on effective address and can only occur if MSR$_{DS}$=0 11 - DAC3 debug events are based on effective address and can only occur if MSR$_{DS}$=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 - DAC4 debug events not affected by MSR$_{PR}$ 01 - Reserved 10 - DAC4 debug events can only occur if MSR$_{PR}$=0 (Supervisor mode) 11 - DAC4 debug events can only occur if MSR$_{PR}$=1. (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 - DAC4 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC4 debug events are based on effective address and can only occur if MSR$_{DS}$=0

Table continues on the next page...

Table 60-13. DBCR8 Bit Definitions (continued)

Bit(s)	Name	Description
		11 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC34M	Data Address Compare 3/4 Mode 00 - Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 - Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 - Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 - Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 - No effect 1 - DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 - No effect 1 - DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

60.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr***DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled (DBCR0$_{IDM}$=1). When debug interrupts are enabled (MSR$_{DE}$=1 DBCR0$_{IDM}$=1 and EDBCR0$_{EDM}$=0, or MSR$_{DE}$=1, DBCR0$_{IDM}$=1, EDBCR0$_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, (EDBCR0$_{EDM}$=1 and EDBRAC0$_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in the following figure.

IDE	UDE	MRR		ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST			0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 60-11. DBSR Register

The following table provides bit definitions for the Debug Status Register.

Table 60-14. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR$_{DE}$=0 and DBCR0$_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 - No reset occurred since these bits were last cleared by software 01 - A hard reset occurred since these bits were last cleared by software 10 - Reserved 11 - Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event

Table continues on the next page...

Table 60-14. DBSR Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a Power ISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

60.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $\mathrm{DBSR}_{\mathrm{DAC\{R,W\}}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $\mathrm{DBSR}_{\mathrm{DAC\{R,W\}}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in the following table.

Data Effective Address
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 600; Read/Write; Reset - unaffected

Figure 60-12. DDEAR Register

60.3.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $\mathrm{EDBCR0}_{\mathrm{EDM}}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $\mathrm{EDBCR0}_{\mathrm{EDM}}$=1. EDBRAC0 settings are ignored when $\mathrm{EDBCR0}_{\mathrm{EDM}}$=0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $\mathrm{DBCR0}_{\mathrm{IDM}}$=1 update DBSR, causing debug interrupts to occur if $\mathrm{MSR}_{\mathrm{DE}}$=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled ($\mathrm{MSR}_{\mathrm{DE}}$=1 $\mathrm{DBCR0}_{\mathrm{IDM}}$=1 and $\mathrm{EDBCR0}_{\mathrm{EDM}}$=0, or $\mathrm{MSR}_{\mathrm{DE}}$=1, $\mathrm{DBCR0}_{\mathrm{IDM}}$=1 and $\mathrm{EDBCR0}_{\mathrm{EDM}}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled (EDBCR0$_{EDM}$=1). When EDBCR0$_{EDM}$=1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If EDBCR0$_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in the following figure.

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	DNI	DQM	0		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b,** reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 60-13. EDBRAC0 Register

Table 60-14 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 60-15. EDBRAC0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control 0 - Internal Debug mode may not be enabled by software. DBCR0$_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 - Internal Debug mode may be enabled by software. DBCR0$_{IDM}$ is owned by software. DBCR0$_{IDM}$ is software readable/writable. When EDBRAC0$_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.

Table continues on the next page...

Table 60-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
2	RST	Reset Field Control 0 - $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 - $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 - Event owned by hardware debug. 1 - Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 - Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 - Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 - Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 - Event owned by software debug. $DBCR0_{TRA}$P is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 - Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 - Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 - Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 - Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields.

Table continues on the next page...

Table 60-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Event owned by software debug. DAC1 control fields are software readable/writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 - Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 - Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 - Event owned by software debug. DAC2 control fields are software readable/writable.
15	—	Reserved
16	RET	Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $\text{DBCR0}_{\text{RET}}$ field. 1 - Event owned by software debug. $\text{DBCR0}_{\text{RET}}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 - Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 - Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 - Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 - Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $\text{DBCR0}_{\text{DEVT1}}$ field. 1 - Event owned by software debug. $\text{DBCR0}_{\text{DEVT1}}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $\text{DBCR0}_{\text{DEVT2}}$ field. 1 - Event owned by software debug. $\text{DBCR0}_{\text{DEVT2}}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event

Table continues on the next page...

Table 60-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Event owned by hardware debug. No **mtspr** access by software to the PMRs. Performance monitor interrupts set EDBSR0$_{PMI}$ regardless of the setting of PMGC0$_{UDI}$ 1 - Event owned by software debug. PMRs are software readable/writable. **Note:** this bit is reset to '1'.
24	MPU	Memory Protection Unit Debug Event 0 - Event owned by hardware debug. No **mpuwe** access by software to region descriptors which have the DEBUG control bit set to '1' or to the MPU0CSR0$_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set EDBSR0$_{MPU}$, and if not masked by EDBSRMSK0$_{MPU}$, one of EDBSR0$_{IAC}$, EDBSR0$_{DACR}$, or EDBSR0$_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 - Event owned by software debug. All region descriptors and the MPU0CSR0$_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** this bit is reset to '1'.
25	CIRPT	Critical Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DBCR0$_{CIRPT}$ field. 1 - Event owned by software debug. DBCR0$_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DBCR0$_{CRET}$ field. 1 - Event owned by software debug. DBCR0$_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 - DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 - DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt (DBCR0$_{IDM}$=1 and MSR$_{DE}$=1) or a nop (DBCR0$_{IDM}$=0 or MSR$_{DE}$=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 - DEVENT$_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to DEVENTD$_{QTAG}$ field or DDAM register. Attempted access by software is ignored. 1 - DEVENT$_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to DEVENT$_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 60-16 shows which resources are controlled by EDBRAC0 settings.

Table 60-16. EDBRAC0 Resource Control

EDBCR0$_{EDM}$	EDBRAC0$_{IDM}$	EDBRAC0$_{RST}$	EDBRAC0$_{UDE}$	EDBRAC0$_{ICMP}$	EDBRAC0$_{BRT}$	EDBRAC0$_{IRPT}$	EDBRAC0$_{TRAP}$	EDBRAC0$_{IAC1}$	EDBRAC0$_{IAC2}$	EDBRAC0$_{IAC3}$	EDBRAC0$_{IAC4}$	EDBRAC0$_{IAC5}$	EDBRAC0$_{IAC6}$	EDBRAC0$_{IAC7}$	EDBRAC0$_{IAC8}$	EDBRAC0$_{DAC1}$	EDBRAC0$_{DAC34}$	EDBRAC0$_{DAC2}$	EDBRAC0$_{RET}$	EDBRAC0$_{DEVT1}$	EDBRAC0$_{DEVT2}$	EDBRAC0$_{PMI}$	EDBRAC0$_{MPU}$	EDBRAC0$_{CIRT}$	EDBRAC0$_{CRET}$	EDBRAC0$_{DNI}$	EDBRAC0$_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR0$_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR0$_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR0$_{UDE}$
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR0$_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR0$_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR0$_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR0$_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, DBCR0$_{IAC1}$, DBCR1$_{IAC1US,IAC1ER}$, DBCR6$_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, DBCR0$_{IAC2}$, DBCR1$_{IAC2US,IAC2ER}$, DBCR6$_{IAC2XM}$
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR1$_{IAC12M}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, DBCR0$_{IAC3}$, DBCR1$_{IAC3US,IAC3ER}$, DBCR6$_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, DBCR0$_{IAC4}$, DBCR1$_{IAC4US,IAC4ER}$, DBCR6$_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR1$_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, DBCR0$_{IAC5}$, DBCR5$_{IAC5US,IAC5ER}$, DBCR6$_{IAC5XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, DBCR0$_{IAC6}$,

Table continues on the next page...

Table 60-16. EDBRAC0 Resource Control (continued)

EDBCR0$_{EDM}$	EDBRAC0$_{IDM}$	EDBRAC0$_{RST}$	EDBRAC0$_{UDE}$	EDBRAC0$_{ICMP}$	EDBRAC0$_{BRT}$	EDBRAC0$_{IRPT}$	EDBRAC0$_{TRAP}$	EDBRAC0$_{IAC1}$	EDBRAC0$_{IAC2}$	EDBRAC0$_{IAC3}$	EDBRAC0$_{IAC4}$	EDBRAC0$_{IAC5}$	EDBRAC0$_{IAC6}$	EDBRAC0$_{IAC7}$	EDBRAC0$_{IAC8}$	EDBRAC0$_{DAC1}$	EDBRAC0$_{DAC34}$	EDBRAC0$_{DAC2}$	EDBRAC0$_{RET}$	EDBRAC0$_{DEVT1}$	EDBRAC0$_{DEVT2}$	EDBRAC0$_{PMI}$	EDBRAC0$_{MPU}$	EDBRAC0$_{CIRT}$	EDBRAC0$_{CRET}$	EDBRAC0$_{DNI}$	EDBRAC0$_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
																												DBCR5$_{IAC6US,IAC6ER}$, DBCR6$_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DBCR5$_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7, DBCR0$_{IAC7}$, DBCR5$_{IAC7US,IAC7ER}$, DBCR6$_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, DBCR0$_{IAC8}$, DBCR5$_{IAC8US,IAC8ER}$, DBCR6$_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	DBCR5$_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U DBCR0$_{DAC1}$, DBCR2$_{DAC1US,DAC1ER}$, DBCR2$_{DVC1M,DVC1BE}$ DBCR4$_{DVC1C,DAC1XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4 DBCR7$_{DAC\{3,4\},DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH}$ DBCR8$_{DAC\{3,4\}US, DAC\{3,4\}ER, DAC\{3,4\}M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U DBCR0$_{DAC2}$, DBCR2$_{DAC2US,DAC2ER}$, DBCR2$_{DVC2M,DVC2BE}$ DBCR4$_{DVC2C,DAC2XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Table continues on the next page...

Table 60-16. EDBRAC0 Resource Control (continued)

EDBCR0$_{EDM}$	EDBRAC0$_{IDM}$	EDBRAC0$_{RST}$	EDBRAC0$_{UDE}$	EDBRAC0$_{ICMP}$	EDBRAC0$_{BRT}$	EDBRAC0$_{IRPT}$	EDBRAC0$_{TRAP}$	EDBRAC0$_{IAC1}$	EDBRAC0$_{IAC2}$	EDBRAC0$_{IAC3}$	EDBRAC0$_{IAC4}$	EDBRAC0$_{IAC5}$	EDBRAC0$_{IAC6}$	EDBRAC0$_{IAC7}$	EDBRAC0$_{IAC8}$	EDBRAC0$_{DAC1}$	EDBRAC0$_{DAC34}$	EDBRAC0$_{DAC2}$	EDBRAC0$_{RET}$	EDBRAC0$_{DEVT1}$	EDBRAC0$_{DEVT2}$	EDBRAC0$_{PMI}$	EDBRAC0$_{MPU}$	EDBRAC0$_{CIRT}$	EDBRAC0$_{CRET}$	EDBRAC0$_{DNI}$	EDBRAC0$_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DBCR8$_{DAC3LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DBCR8$_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	DBCR0$_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	DBCR0$_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	DBCR0$_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	MPU entries with DEBUG bit set, MPU0CSR0$_{DRDEN, DWDEN,IDEN}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	DBCR0$_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	DBCR0$_{CRET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	1	-	
1	-[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	DEVENT$_{DQTAG}$, DDAM

[1] Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when EDBCR0$_{EDM}$=1. Only software-owned bits or fields as shown in Table 60-16 are affected in this case, except that DBCR0$_{RST}$ and DBSR$_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of EDBCR0$_{EDM}$ or EDBRAC0.

60.3.4 Debug Event Select Register (DEVENT)

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The DEVENT$_{\text{DEVNT}}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z4251n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in the following figure.

DQTAG	0	DEVNT
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31

SPR - 975; Reset[1] - 0x0

Figure 60-14. DEVENT Register

Note

[1] Reset by processor reset **p_reset_b** if EDBCR0$_{\text{EDM}}$=0, as well as unconditionally by **m_por**. If EDBCR0$_{\text{EDM}}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

The following table provides bit definitions for the Debug Event Register.

Table 60-17. DEVENT Bit Definitions

Bit(s)	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 - No signal is asserted xxxxxxx1 - **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock xxxxxx1x - **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock xxxxx1xx - **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock xxxx1xxx - **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock xxx1xxxx - **p_devnt_out[4]** is asserted for one clock xx1xxxxx - **p_devnt_out[5]** is asserted for one clock

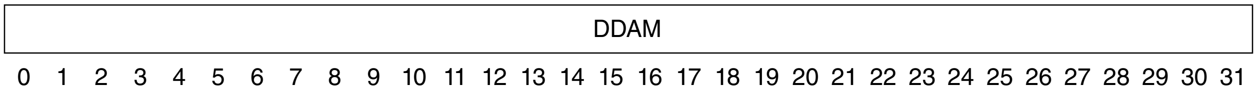
Table 60-17. DEVENT Bit Definitions

Bit(s)	Name	Description
		x1xxxxxx - **p_devnt_out[6]** is asserted for one clock 1xxxxxxx - **p_devnt_out[7]** is asserted for one clock

60.3.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Core (e200z4251n3) Nexus 3 Module chapter for details.

The DDAM register is shown in the following figure.



SPR - 576; Reset[1] - 0x0

Figure 60-15. DDAM Register

Note

[1] Reset by processor reset **p_reset_b** if EDBCR0$_{EDM}$=0, as well as unconditionally by **m_por**. If EDBCR0$_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for the Debug Data Acquisition Message Register.

Table 60-18. DDAM Bit Definitions

Bit(s)	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

60.4 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in Table 60-9. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $\mathrm{DBCR4}_{\mathrm{DAC1CFG}}$ field to '1000' or the $\mathrm{DBCR8}_{\mathrm{DAC3CFG}}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $\mathrm{DBCR4}_{\mathrm{DAC1CFG}}$ field is set to '1000' with $\mathrm{DBCR2}_{\mathrm{DAC1US}}$ set to '00' or '10', or the $\mathrm{DBCR7}_{\mathrm{DAC3CFG}}$ field is set to '1000' with $\mathrm{DBCR8}_{\mathrm{DAC3US}}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $\mathrm{DBCR4}_{\mathrm{DAC1CFG}}$ field is set to '1000' with $\mathrm{DBCR2}_{\mathrm{DAC1US}}$ set to '00' or '01', or the $\mathrm{DBCR7}_{\mathrm{DAC3CFG}}$ field is set to '1000' with $\mathrm{DBCR8}_{\mathrm{DAC3US}}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $\mathrm{EDBSR0}_{\mathrm{IDE}}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in Table 60-9 and Table 60-12 in Debug Control and Status registers. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

60.5 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of Software Debug Events and Exceptions refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note

On the initial setting of EDBCR0$_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After EDBCR0$_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR$_{DR}$ control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

Note

It is intended for the CPU to remain in external debug mode (EDBCR0$_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR$_{DR}$, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note

EDBCR0$_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if EDBCR0$_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all *hardware* debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the EDBCR0$_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

60.5.1 External Debug Registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

60.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $\text{EDBCR0}_{\text{EDM}}$ is reflected as a read-only bit in $\text{DBCR0}_{\text{EDM}}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in the following figure.

EDM	DNH_EN	0	DNI_CTL	0
0	1	2 3 4 5 6	7	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Reset[1] - 0x0

Figure 60-16. EDBCR0 Register

Note

[1] EDBCRO is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b.**

The following table provides bit definitions for External Debug Control Register 0.

Table 60-19. EDBCR0 Bit Definitions

Bit(s)	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0 0 - External debug mode disabled. Internal debug events not mapped into external debug events. 1 - External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1	DNH_EN	**dnh** Instruction Enable 0 - execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 - execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	---	Reserved
7	DNI_CTL	**dni** Instruction Control 0 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSRDE=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1
8:31	---	Reserved

60.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in the following figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST			0

Read/Write; Reset[1] - 0x0000_0000

Figure 60-17. EDBSR0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while EDBCRO$_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register 0.

Table 60-20. EDBSR0 Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if EDBCR0$_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved

Table continues on the next page...

Table 60-20. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a Power ISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH\_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

60.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in the following figure.

0	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	0				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Read/Write; Reset[1] - 0x0000_0000

Figure 60-18. EDBSRMSK0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register Mask 0.

Table 60-21. EDBSRMSK0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$

Table continues on the next page...

Table 60-21. EDBSRMSK0 Bit Definitions (continued)

Bit(s)	Name	Description
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

60.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the EDBSR0$_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in the following figure.

Data Effective Address

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Read/Write; Reset - unaffected

Figure 60-19. EDDEAR Register

60.5.2 OnCE Introduction

The e200z4251n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z4251n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/ software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

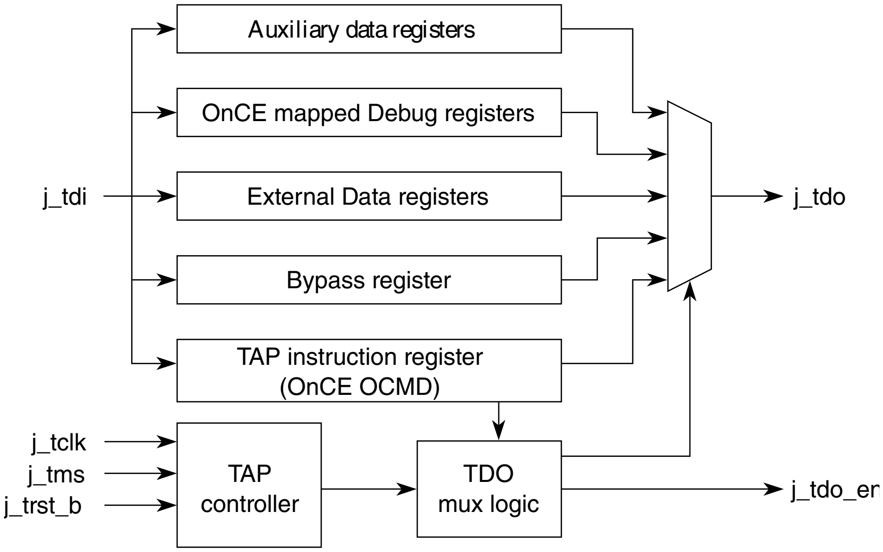
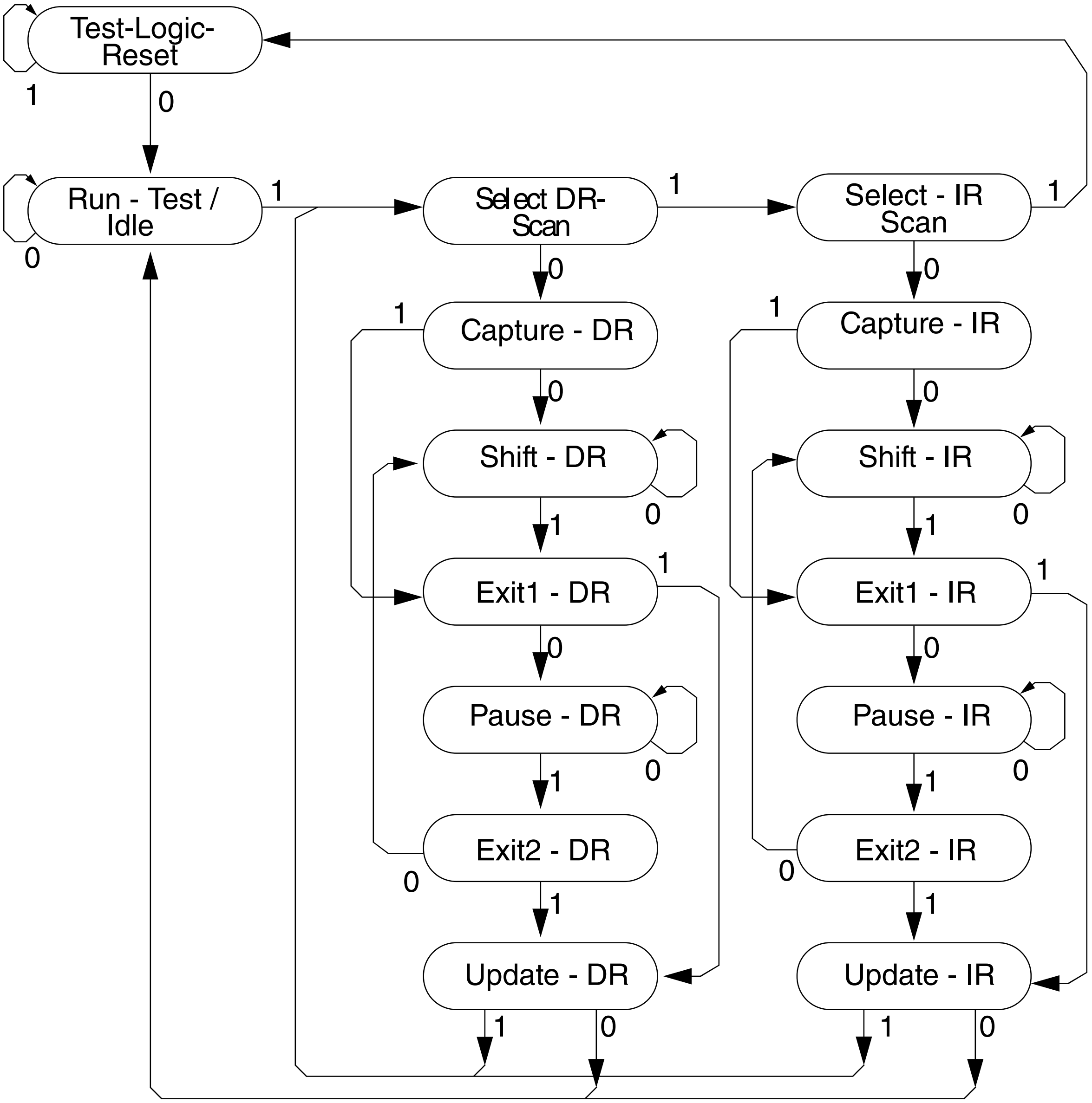


Figure 60-20. OnCE TAP Controller and Registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting EDBCR0$_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), OnCE Status Register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See OnCE Command Register (OCMD)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

60.5.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

The following table details the primary JTAG/OnCE interface signals.

Table 60-22. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer

[1] j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z4251n3) Core Complex Overview.

60.5.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

60.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

60.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

60.5.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

60.5.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

60.5.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks which may need information pertaining to debug operation.

60.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/ Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

60.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z4251n3 interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

60.5.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either "No Register Selected" or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See OnCE Command Register (OCMD) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

60.5.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z4251n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

60.5.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to Table 60-28 for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

60.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates "no register selected". This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with "no register selected", and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

60.5.6 OnCE Controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. The following figure is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

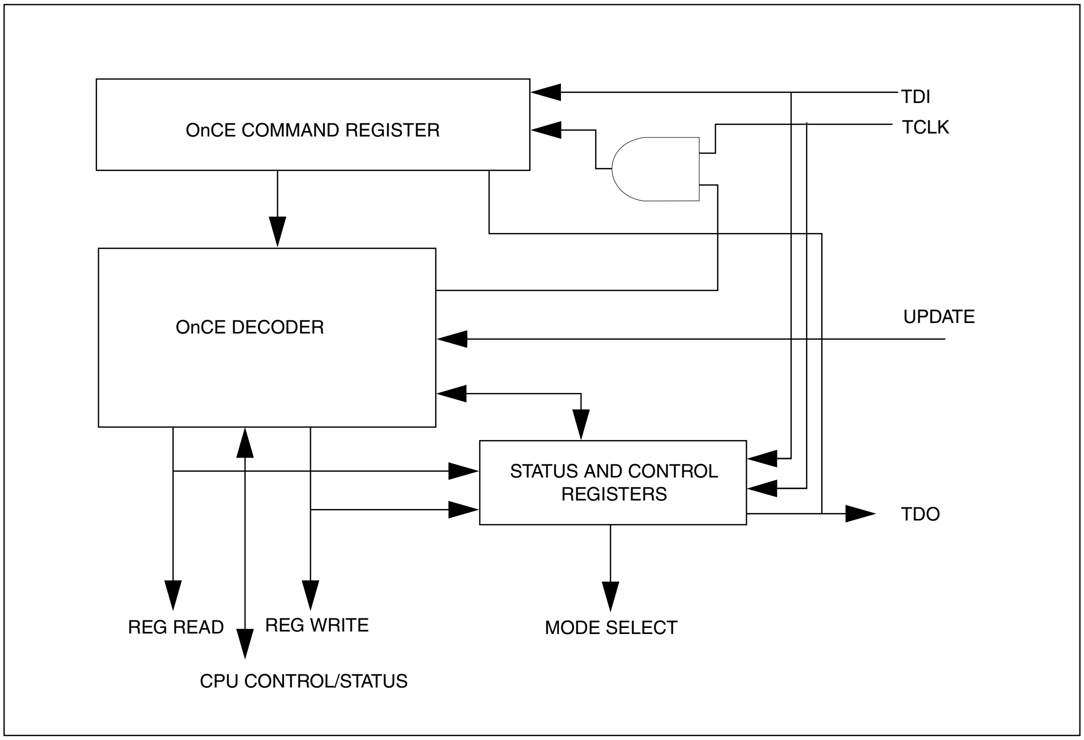


Figure 60-21. OnCE Controller and Serial Interface

60.5.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 60-22. OnCE Status Register

The following table provides bit definitions for the Once Status Register.

Table 60-23. OnCE Status Register Bit Definitions

Bit(s)	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 - Inactive state 1 - Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.

Table continues on the next page...

Table 60-23. OnCE Status Register Bit Definitions (continued)

Bit(s)	Name	Description
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

60.5.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 60-23. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

R/W	GO	EX	RS[0:6]						
0	1	2	3	4	5	6	7	8	9

Reset - 10’b1000000010 on assertion of **j_trst_b** or **m_por** , or while in the Test_Logic_Reset state

Figure 60-23. OnCE Command Register

Table 60-24 provides bit definitions for the Once Command Register.

Table 60-24. OnCE Command Register Bit Definitions

Bit(s)	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 - Write the data associated with the command into the register specified by RS[0:6]

Table continues on the next page...

Table 60-24. OnCE Command Register Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Read the data contained in the register specified by RS[0:6]
1	GO	Go Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The $\mathrm{OSR}_{\mathrm{ERR}}$ bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking. 0 - Inactive (no action taken) 1 - Execute instruction in IR
2	EX	Exit Command Bit 0 - Remain in debug mode 1 - Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $\mathrm{EDBCR0}_{\mathrm{EDM}}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 60-25 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 60-25 indicates the OnCE register addresses.

Table 60-25. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved

Table continues on the next page...

Table 60-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)

Table continues on the next page...

Table 60-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3-Access (<<<put in cross reference to Nexus 3 Module>>>)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE[1]
111 1111	Bypass

[1] Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The $\mathrm{OSR}_{\mathrm{ERR}}$ bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $\mathrm{EDBCR0}_{\mathrm{EDM}}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0EDM. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

60.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z4251n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see MPU Operation During Debug). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in the following figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0								I_DMDIS	0		I_DVLE	I_DI	0			D_DMDIS	0			D_DI	0	D_DG	0						WKUP	FDB	DR

Reset - 0xo000_0000 on **m_por, j_trst_b,** or entering Test_logic_Reset state

Figure 60-24. OnCE Control Register

The following table provides bit definitions for the OnCE Control Register.

Table 60-26. OnCE Control Register Bit Definitions

Bit(s)	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit (I_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug 'VLE' Attribute Bit (I_DVLE) This bit is used to provide the 'VLE' attribute bit to be used during a debug session. Note: this bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug 'I' Attribute Bit (I_DI) This bit is used to provide the 'I' attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit (D_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions

Table continues on the next page...

Table 60-26. OnCE Control Register Bit Definitions (continued)

Bit(s)	Name	Description
		This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
17:19	—	Reserved
20	D_DI	Data Side Debug 'I' Attribute Bit (D_DI) This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug 'G' Attribute Bit (D_DG) This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit (WKUP) This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with EDBCR0$_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 - No Debug Mode request 1 - Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

60.5.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the e200z4251n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z4251n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z4251n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

The following table provides a list of access requirements for OnCE registers.

Table 60-27. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires $\text{EDBCR0}_{\text{EDM}}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*	
DBCR0	Y	Y	Y	N	*[1]	*$\text{EDBCR0}_{\text{EDM}}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*[1]	
DEVENT	Y	Y	Y	N	*[1]	
EDBRAC0 (DBERC0)	Y	N	Y	N	*[1]	
DBSR	Y	Y	Y	N	*[1]	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	

Table continues on the next page...

Table 60-27. OnCE Register Access Requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*[1]	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.

60.5.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See OnCE Enable (jd_en_once).

60.5.8.1 External Debug Request During RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b,** and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z4251n3 core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z4251n3 core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.

60.5.8.2 Debug Request During RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

60.5.8.3 Debug Request During Normal Activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

60.5.8.4 Debug Request During Waiting, Halted or Stopped State

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to $\mathrm{OCR}_{\mathrm{DR}}$ being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** outputs during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

60.5.8.5 Software Request During Normal Activity

Upon executing a '**bkpt**' pseudo-instruction (for e200z4251n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and EDBCR0$_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.

60.5.8.6 Debug Notify Halt Instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by EDBCR0$_{DNH\_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on EDBCR0$_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z4251n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

60.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. The following figure shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

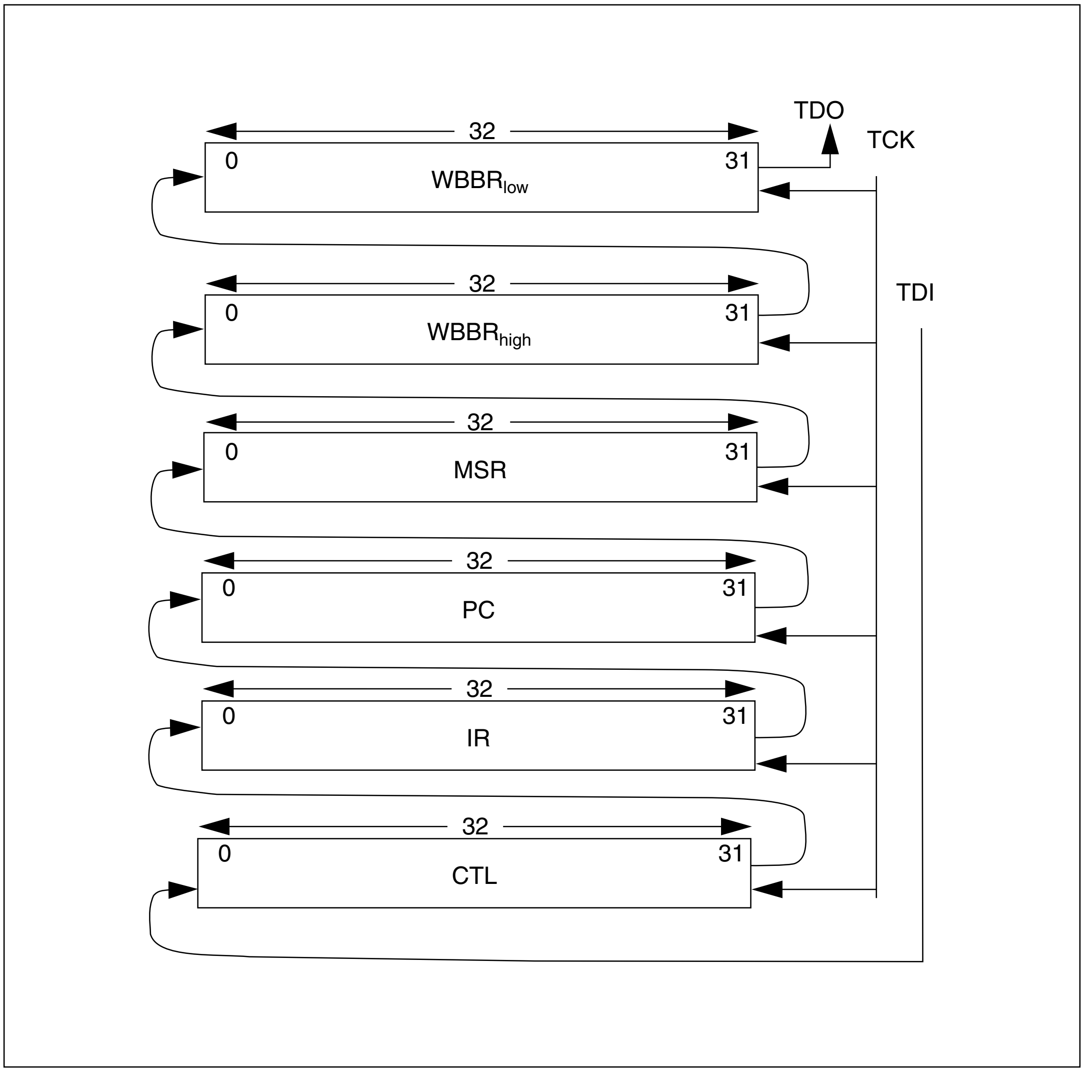


Figure 60-25. CPU Scan Chain Register (CPUSCR)

60.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

60.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

*										IRSTAT14	IRSTAT13	IRSTAT12	IRSTAT11	IRSTAT10	WAITING	PCOFST				PCINV	FFRA	IRSTAT0	IRSTAT1	IRSTAT2	IRSTAT3	IRSTAT4	IRSTAT5	IRSTAT6	IRSTAT7	IRSTAT8	IRSTAT9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Figure 60-26. Control State Register (CTL)

WAITING — WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0- CPU was not in the waiting state when debug mode was entered

1- CPU was in the waiting state when debug mode was entered

PCOFST — PC Offset Field

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 - No correction required.

0001 - Subtract 0x04 from PC.

0010 - Subtract 0x08 from PC.

0011 - Subtract 0x0C from PC.

0100 - Subtract 0x10 from PC.

0101 - Subtract 0x14 from PC.

all other encodings are reserved

* — Internal State Bits

Table continues on the next page...

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See OnCE Debug Output (jd_debug_b), which is not part of the normal program execution flow, these bits should be set to a 0.

PCINV — PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0= No error condition exists.

1= Error condition exists. PC and IR are corrupted.

FFRA— Feed Forward RA Operand Bit

This control bit causes the content of the WBBR$_{lo}$ to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.For most LSP instructions using rA||rB as a 64-bit source operand, WBBR$_{hi, lo}$ is used to supply the 64-bit source value.

This allows the debug firmware to update processor registers — initialize the WBBR$_{lo}$ with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register.

Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.

0= No action.

1= Content of WBBR$_{lo}$or WBBR$_{hi,lo}$used as operand value

IRStat0 — IR Status Bit 0

This control bit indicates a TEA status for the IR.

0= No TEA occurred on the fetch of this instruction.

1= TEA occurred on the fetch of this instruction.

IRStat1 — IR Status Bit 1

This control bit is reserved.

IRStat2 — IR Status Bit 2

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0= No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 1 occurred on the fetch of this instruction.

IRStat3 — IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0= No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 2 event occurred on the fetch of this instruction.

IRStat4 — IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0= No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 3 occurred on the fetch of this instruction.

IRStat5 — IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

Table continues on the next page...

0= No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 4 event occurred on the fetch of this instruction.

IRStat6 — IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0= No Parity Error occurred on the fetch of this instruction.

1= Parity Error occurred on the fetch of this instruction from the I-Cache .

IRStat7 — IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0= No Precise External Termination Error occurred on the fetch of this instruction.

1= A Precise External Termination Error occurred on the fetch of this instruction.

IRStat8 — IR Status Bit 8

This control bit indicates the Power ISA VLE status for the IR.

0= IR contains a BookE instruction. (unused encoding)

1= IR contains a Power ISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

- this bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).

IRStat9 — IR Status Bit 9

This control bit is reserved.

IRStat10 — IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0= No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 5 event occurred on the fetch of this instruction.

IRStat11 — IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0= No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 6 event occurred on the fetch of this instruction.

IRStat12 — IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

0= No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 7 event occurred on the fetch of this instruction.

IRStat13 — IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0= No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 8 event occurred on the fetch of this instruction.

IRStat14 — IR Status Bit 14

This control bit indicates an MPU Instruction Address Compare event status for the IR.

0= No MPU Instruction Address Compare event occurred on the fetch of this instruction.

1= An MPU Instruction Address Compare event occurred on the fetch of this instruction.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

60.5.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

60.5.9.4 Write-Back Bus Register (WBBR$_{low}$, WBBR$_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. WBBR$_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. For LSP instructions which generate 64-bit results, WBBR$_{low}$ holds the low-order 32 bits of the result. WBBR$_{high}$ holds the updated effective address calculated by a load with update instruction. For LSP instructions which generate 64-bit results, WBBR$_{high}$ holds the high-order 32 bits of the result. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into WBBR$_{low}$. The contents of WBBR$_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the WBBR$_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to Control State Register (CTL) for more detail on the CTL$_{FFRA}$ bit.)

WBBR$_{low}$ and WBBR$_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

60.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

60.5.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, "asynchronous" interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

60.5.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

60.6 Watchpoint Support

e200z4251n3 supports the generation and signalling of watchpoints when operating in internal debug mode ($\mathrm{DBCR0}_{\mathrm{IDM}}$=1) or in external debug mode ($\mathrm{EDBCR0}_{\mathrm{EDM}}$=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $\mathrm{EDBCR0}_{\mathrm{EDM}}$ or $\mathrm{DBCR0}_{\mathrm{IDM}}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. $\mathrm{MSR}_{\mathrm{DE}}$ has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31]**) for various development control purposes. (See the "Watchpoint Trace Messaging" section in the Core (e200z4251n3) Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

The following table shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 60-28. Watchpoint Output Signal Assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on DBCR4$_{\text{DAC1CFG}}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on DBCR4$_{\text{DAC2CFG}}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on DBCR7$_{\text{DAC3CFG}}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on DBCR7$_{\text{DAC4CFG}}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint

Table continues on the next page...

Table 60-28. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
		Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever **p_pm_event** transitions from a '0' to a '1' while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on $\text{PMLCa0}_{\text{PMP}}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on $\text{PMLCa1}_{\text{PMP}}$
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on $\text{PMLCa2}_{\text{PMP}}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on $\text{PMLCa3}_{\text{PMP}}$
jd_watchpt[27]	MPU	Memory Protection Unit watchpoint Asserted whenever a the MPU generates a debug event based on region descriptor matches with DEBUG control enabled.
jd_watchpt[28]	TRAP	TRAP watchpoint

Table continues on the next page...

Table 60-28. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
		Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

60.7 MPU Operation During Debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

60.8 Cache Array Access During Debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

60.9 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which $\mathrm{OCR}_{\mathrm{DR}}$, $\mathrm{OCR}_{\mathrm{WKUP}}$, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode.
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $\mathrm{EDBCR0}_{\mathrm{EDM}}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the PCU has re-entered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug Mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should re-write the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note

These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

Chapter 61
JTAG Controller (JTAGC)

61.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

61.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as Register description for more information about the JTAGC registers.

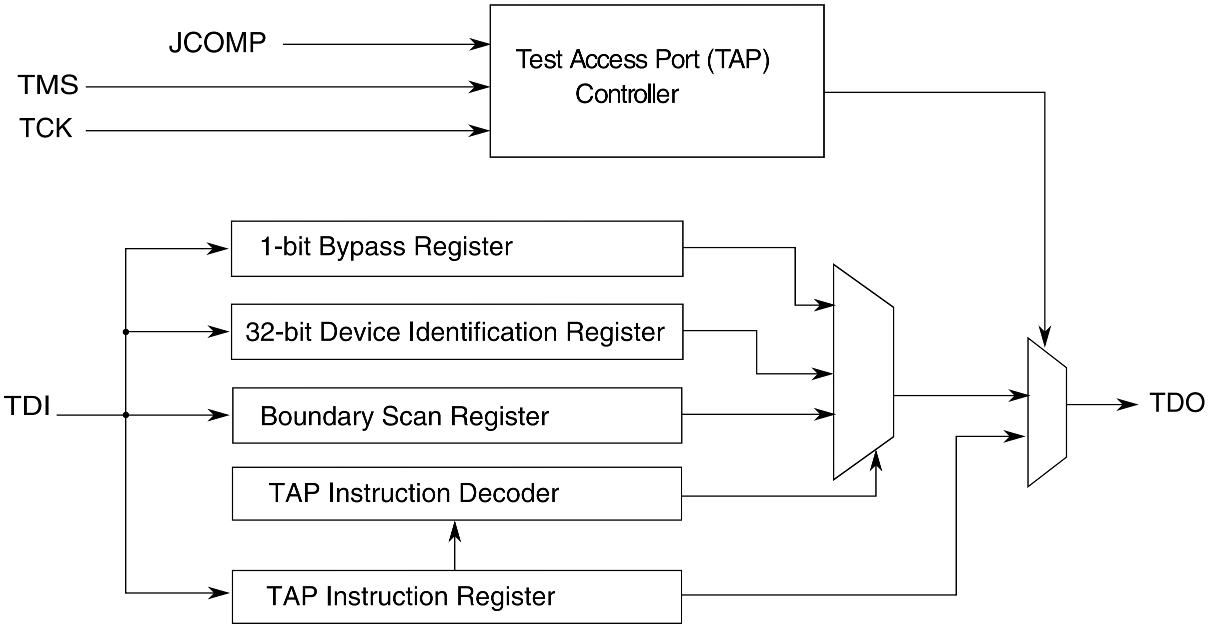


Figure 61-1. JTAG (IEEE 1149.1) block diagram

61.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to Table 61-5 for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- TEST_CTRL register
- JTAG_PASSWORD register
- Bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

61.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

61.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

61.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in JTAGC block instructions.

61.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

61.2 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 61-1. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Up
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

61.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

61.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

61.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

61.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

61.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

61.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

61.3.1 Instruction register

The JTAGC block uses a 5-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 00001b , making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	4	3	2	1	0
R	0	0	0	0	1
W	Instruction Code				
Reset:	0	0	0	0	1

Figure 61-2. Instruction register

61.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

61.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

<table>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>R</td><td colspan="4">Part Revision Number</td><td colspan="6">Design Center</td><td colspan="6">Part Identification Number</td></tr>
<tr><td>W</td><td colspan="4"></td><td colspan="6"></td><td colspan="6"></td></tr>
<tr><td>Reset</td><td colspan="4">PRN</td><td colspan="6">DC</td><td colspan="6">PIN</td></tr>
<tr><td></td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>R</td><td colspan="4">Part Identification Number</td><td colspan="11">Manufacturer Identity Code</td><td>1</td></tr>
<tr><td>W</td><td colspan="4"></td><td colspan="11"></td><td></td></tr>
<tr><td>Reset</td><td colspan="4">PIN (contd.)</td><td colspan="11">MIC</td><td>1</td></tr>
</table>

The following table describes the device identification register functions.

Table 61-2. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is as mentioned in Table - "PRN for existing Masksets" given in Debug section of "Chip configuration" chapter..
DC	Design Center. Indicates the design center. Value is 0x26.
PIN	Part Identification Number. Contains the part number of the device. 0x345.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

61.3.4 TEST_CTRL Register

The TEST_CTRL register is a 17-bit shift register path from TDI to TDO selected when the ENABLE_TEST_CTRL instruction is active. The TEST_CTRL register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state.

	16	...	2	1	0
R	TEST_CTRL				
W					
Reset:	*[1]	*	*	*	*

[1] The reset value of TEST_CTRL is 0x5D

The following table describes the TEST_CTRL register functions.

Table 61-3. TEST_CTRL register field descriptions

Field	Description
TEST_CTRL	Test Control. Used to control chiptop test functions.

61.3.5 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 65-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 65'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.

	64	...	2	1	0
R	JTAG_PASSWORD				
W					
Reset:	*[1]	*	*	*	*

[1] The reset value of JTAG_PASSWORD is 65 'b0.

[1] The reset value of JTAG_PASSWORD is 65 'b0.

The following table describes the JTAG_PASSWORD register functions.

Table 61-4. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

61.3.6 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

61.4 Functional description

This section explains the JTAGC functional description.

61.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

61.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to ACCESS_AUX_x instructions.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

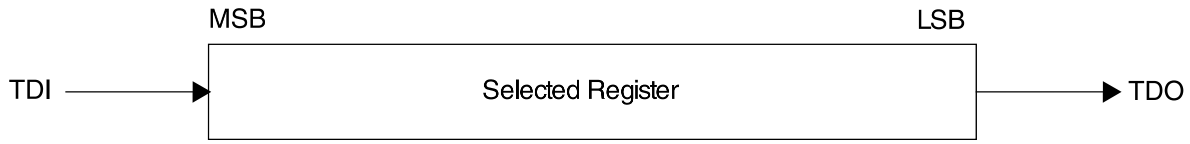
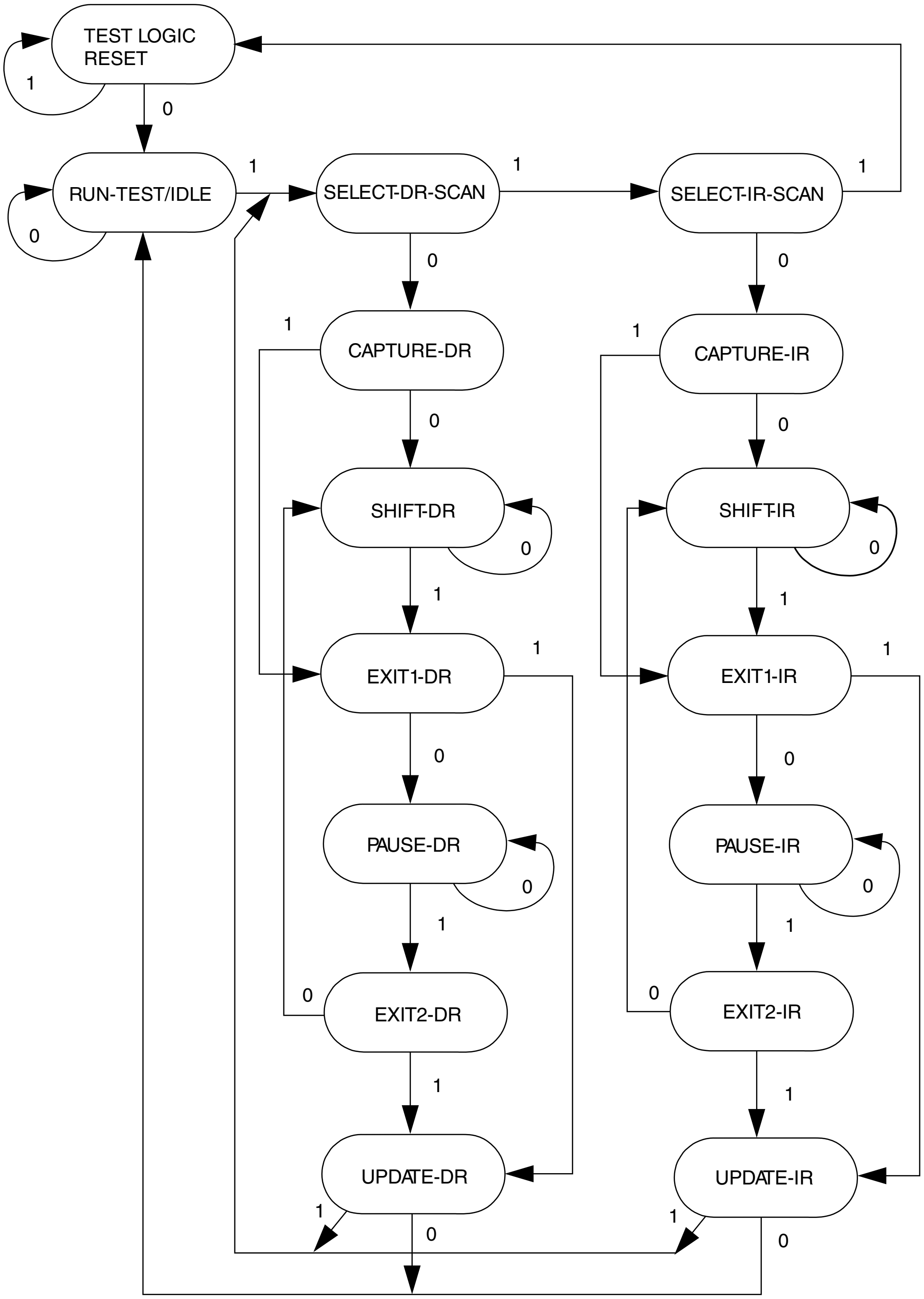


Figure 61-3. Shifting data through a register

61.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 61-4. IEEE 1149.1-2001 TAP controller finite state machine

61.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

61.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

61.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 61-5. General 5-bit JTAG instructions

Instruction	Code[4:0]	Instruction Summary
IDCODE	00001	Selects device identification register for shift
SAMPLE/PRELOAD	00010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	00011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	00100	Selects boundary scan register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	00101	Intended for factory debug only
ENABLE_TEST_CTRL	00110	Selects TEST_CTRL register
ENABLE_JTAG_PASSWORD	00111	Selects JTAG_PASSWORD register
HIGHZ	01001	Selects bypass register and three-states all output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	01010	Intended for factory debug only
CLAMP	01100	Selects bypass register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.

Table continues on the next page...

Table 61-5. General 5-bit JTAG instructions (continued)

Instruction	Code[4:0]	Instruction Summary
Factory debug reserved	01110	Intended for factory debug only
Factory debug reserved	01111	Intended for factory debug only
BYPASS	11111	Selects bypass register for data operations
Reserved[1]	All other opcodes	Decoded to select bypass register

[1] The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

The ACCESS_AUX instructions are described in the chip configuration debug information.

61.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

61.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

61.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

61.4.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

61.4.4.5 ENABLE_TEST_CTRL instruction

The ENABLE_TEST_CTRL instruction selects the TEST_CTRL register for connection as the shift path between TDI and TDO.

61.4.4.6 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

61.4.4.7 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

61.4.4.8 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

61.4.4.9 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

61.4.4.10 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

61.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

61.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller.
2. Load the appropriate instruction for the test or action to be performed.

Chapter 62
Nexus Module

62.1 Introduction

The e200z4251n3Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

62.1.1 General description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

62.1.2 Terms and definitions

The following table contains a set of terms and definitions associated with the Nexus 3 module.

Table 62-1. Terms and definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	"System-on-a-Chip". SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase "according to the standard" is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

62.1.3 Feature list

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- Auxiliary interface for higher data input
    - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
    - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
    - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
    - One (1) Watchpoint Event output pin (**evto_b**)
    - One (1) Event In pin (**nex_evti_b**)
    - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)
- Indicates to the FCCU whether it becomes active during functional mode.

Note

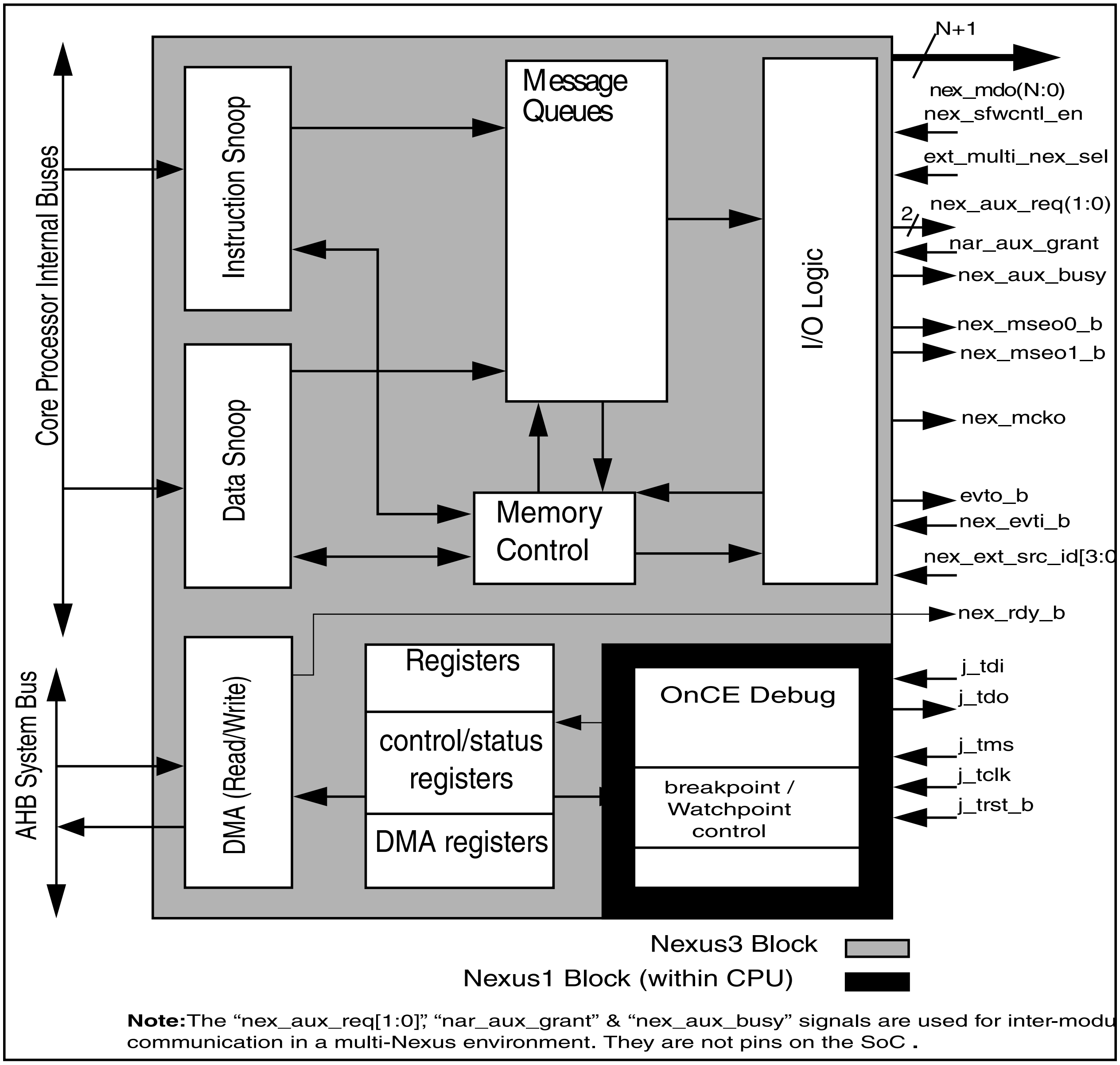
For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (at the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note

The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

62.1.4 Functional block diagram



Note: The “nex_aux_req[1:0]”, “nar_aux_grant” & “nex_aux_busy” signals are used for inter-modu communication in a multi-Nexus environment. They are not pins on the SoC .

Figure 62-1. Nexus 3 functional block diagram

62.2 Enabling Nexus 3 operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

62.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

62.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in Table 62-2.

Table 62-2. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 62-7)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table continues on the next page...

Table 62-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 62-7)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from DEVENT$_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch Message w/Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)

Table continues on the next page...

Table 62-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 62-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 62-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 62-5) - indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section Resource Full Messages.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions

Table continues on the next page...

Table 62-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	32	HIST	variable	branch / predicate instruction history (see Branch Trace Messaging types)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Branch Trace Messaging types.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 62-6)
	2	2	CDF	fixed	# fields of information in CDATA. 00 - reserved, 01 - one field (CDATA1) (reserved), 10 - two fields (CDATA1 + CDATA2), 11 - three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 - [branch / predicate instruction history] (See Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2- PID info (See Program Correlation Messages.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)

Note

Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/ Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 62-3 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 62-3. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 62-4 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 62-4. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 - 1111	Reserved

Table 62-5 shows the encodings used for resource codes for certain messages.

Table 62-5. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 62-6 shows the event code encodings used for certain messages.

Table 62-6. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace

Table continues on the next page...

Table 62-6. Event Code (EVCODE) Encoding (TCODE = 33) (continued)

Event Code	Description
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 62-7 shows the data trace size encodings used for certain messages.

Table 62-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 - no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

62.4 Nexus 3 Programmer's model

This section describes the Nexus 3 programmer's model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See Nexus 3 Register Access via JTAG/OnCE and Nexus 3 Register Access via Software for details on Nexus 3 register access.

Note

Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

The following table details the register map for the Nexus 3 module.

Table 62-8. Nexus 3 register map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Client Select Control (CSC)	0x1	R	0x02	-	
Port Configuration Register (PCR)[3]	PCR_INDEX[2]	R/W	-	-	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	-
Reserved	0x8	R/W	0x18	0x19	
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	-
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	-
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 -> 0x2F	-	0x28->0x5E	0x29->5F	
Development Status (DS)	0x30	R	0x60	-	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 -> 0x3F	-	0x72->0x7E	0x73->7F	

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.

3. The "PCR_INDEX" is a parameter determined by the SoC.

62.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.

7	6	5	4	3	2	1	0
Reserved				CS			

Nexus Reg# - 0x1; Read-only; Reset - 0x0

Figure 62-2. Client Select Control (CSC) register

Table 62-9. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx - Nexus client (SoC level)

62.4.2 Port Configuration Register (PCR) - reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPC	0	MCK_EN	MCK_DIV			0																									

Nexus Reg# - PCR_INDEX; Read/Write; Reset - 0x0

Figure 62-3. Port Configuration Register

Table 62-10. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality

Table continues on the next page...

Table 62-10. PCR field descriptions (continued)

Bit	Name	Description
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note

The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

62.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in Figure 62-4 and its fields are described in Table 62-11.

OPC	MCK_DIV		0	PTM	0		OTS	0									POTD	TSEN		EOC		EIC		0		TM					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x2; DCR - 368; Read/Write; Reset - 0x0

Figure 62-4. Development Control Register 1

Table 62-11. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 **nex_mcko** is 1x processor clock freq. 01 **nex_mcko** is 1/2x processor clock freq. 10 **nex_mcko** is 1/4x processor clock freq. 11 **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 Program Trace uses traditional Branch Messages 1 Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 PID0 data is transmitted within Ownership Trace Messages 1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 Periodic Ownership Trace message events are enabled 1 Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable - (not implemented, write to 00) 00 Timestamp is disabled
11:10	EOC	EVTO Control 00 **evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 **evto_b** upon entry into Debug Mode 1x Reserved
9:8	EIC	EVTI Control 00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 **nex_evti_b** is used for Debug request 10 **nex_evti_b** is disabled 1X Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode[1] 000000 All Trace Disabled XXXXX1 Ownership Trace enabled XXXX1X Data Trace enabled XXX1XX Program Trace enabled XX1XXX Watchpoint Trace enabled X1XXXX Reserved 1XXXXX Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.
1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.
1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.

Note

> The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

62.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z4251n3) Core Debug Support chapter.

Development Control Register 2 is shown in Figure 62-5 and its fields are described in Table 62-12.

<table>
<tr><td colspan="4">0</td><td colspan="4">0</td><td colspan="4">0</td><td colspan="4">0</td><td colspan="16">EWC</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Nexus Reg# - 0x3; DCR - 369; Read/Write; Reset - 0x0

Figure 62-5. Development Control Register 2

Table 62-12. DC2 field descriptions

Bits	Name	Description
31:16	—	Reserved
15:0	EWC	EVTO Watchpoint Configuration[1] 0000000000000000 No Watchpoints #0-15 trigger **evto_b** XXXXXXXXXXXXXXX1 Watchpoint #0 triggers **evto_b** XXXXXXXXXXXXXX1X Watchpoint #1 triggers **evto_b**

Table 62-12. DC2 field descriptions

Bits	Name	Description
		XXXXXXXXXXXXX1XX Watchpoint #2 triggers **evto_b**
		XXXXXXXXXXXX1XXX Watchpoint #3 triggers **evto_b**
		XXXXXXXXXXX1XXXX Watchpoint #4 triggers **evto_b**
		XXXXXXXXXX1XXXXX Watchpoint #5 triggers **evto_b**
		XXXXXXXXX1XXXXXX Watchpoint #6 triggers **evto_b**
		XXXXXXXX1XXXXXXX Watchpoint #7 triggers **evto_b**
		XXXXXXX1XXXXXXXX Watchpoint #8 triggers **evto_b**
		XXXXXX1XXXXXXXXX Watchpoint #9 triggers **evto_b**
		XXXXX1XXXXXXXXXX Watchpoint #10 triggers **evto_b**
		XXXX1XXXXXXXXXXX Watchpoint #11 triggers **evto_b**
		XXX1XXXXXXXXXXXX Watchpoint #12 triggers **evto_b**
		XX1XXXXXXXXXXXXX Watchpoint #13 triggers **evto_b**
		X1XXXXXXXXXXXXXX Watchpoint #14 triggers **evto_b**
		1XXXXXXXXXXXXXXX Watchpoint #15 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in Figure 62-6 and its fields are described in Table 62-13.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				0				0				0				0		EWC													

Nexus Reg# - 0x4; DCR - 370; Read/Write; Reset - 0x0

Figure 62-6. Development Control 3 (DCR3) register

Table 62-13. DC3 field descriptions

Bits	Name	Description
31:16	—	Reserved
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration[1] 00000000000000 No Watchpoints #16-#29 trigger **evto_b** XXXXXXXXXXXXX1 Watchpoint #16 triggers **evto_b** XXXXXXXXXXXX1X Watchpoint #17 triggers **evto_b** XXXXXXXXXXX1XX Watchpoint #18 triggers **evto_b** XXXXXXXXXX1XXX Watchpoint #19 triggers **evto_b** XXXXXXXXX1XXXX Watchpoint #20 triggers **evto_b** XXXXXXXX1XXXXX Watchpoint #21 triggers **evto_b**

Table 62-13. DC3 field descriptions

Bits	Name	Description
		XXXXXXX1XXXXXX Watchpoint #22 triggers **evto_b** XXXXXX1XXXXXXX Watchpoint #23 triggers **evto_b** XXXXX1XXXXXXXX Watchpoint #24 triggers **evto_b** XXXX1XXXXXXXXX Watchpoint #25 triggers **evto_b** XXX1XXXXXXXXXX Watchpoint #26 triggers **evto_b** XX1XXXXXXXXXXX Watchpoint #27 triggers **evto_b** X1XXXXXXXXXXXX Watchpoint #28 triggers **evto_b** 1XXXXXXXXXXXXX Watchpoint #29 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

62.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in Figure 62-7 and its fields are described in Table 62-14.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTMARK	DTMARK	0														EVCDM															

Nexus Reg# - 0x5; DCR - 371; Read/Write; Reset - 0x0

Figure 62-7. Development Control 4 (DC4) register

Table 62-14. DC4 field descriptions

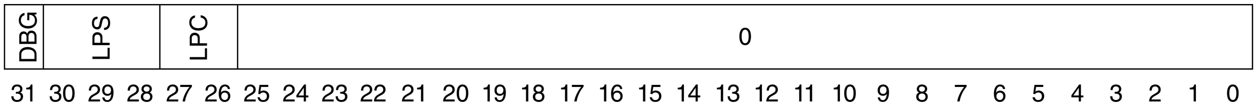
Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	DTMARK Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask

Table 62-14. DC4 field descriptions

Bits	Name	Description
		For implemented EVCODEs. please see Table 62-6. 0000000000000000 No EVCODEs masked for Program Correlation Messages XXXXXXXXXXXXXXX1 EVCODE #0 is masked for Program Correlation Messages XXXXXXXXXXXXXX1X EVCODE #1 is masked for Program Correlation Messages XXXXXXXXXXXXX1XX EVCODE #2 is masked for Program Correlation Messages XXXXXXXXXXXX1XXX EVCODE #3 is masked for Program Correlation Messages XXXXXXXXXXX1XXXX EVCODE #4 is masked for Program Correlation Messages XXXXXXXXXX1XXXXX EVCODE #5 is masked for Program Correlation Messages XXXXXXXXX1XXXXXX EVCODE #6 is masked for Program Correlation Messages XXXXXXXX1XXXXXXX EVCODE #7 is masked for Program Correlation Messages XXXXXXX1XXXXXXXX EVCODE #8 is masked for Program Correlation Messages XXXXXX1XXXXXXXXX EVCODE #9 is masked for Program Correlation Messages XXXXX1XXXXXXXXXX EVCODE #10 is masked for Program Correlation Messages XXXX1XXXXXXXXXXX EVCODE #11 is masked for Program Correlation Messages XXX1XXXXXXXXXXXX EVCODE #12 is masked for Program Correlation Messages XX1XXXXXXXXXXXXX EVCODE #13 is masked for Program Correlation Messages X1XXXXXXXXXXXXXX EVCODE #14 is masked for Program Correlation Messages 1XXXXXXXXXXXXXXX EVCODE #15 is masked for Program Correlation Messages

62.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



Nexus Reg# - 0x4; DCR - 409; Read-only; Reset - 0x0

Figure 62-8. Development Status Register

Table 62-15. DS field description

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode

Table continues on the next page...

Table 62-15. DS field description (continued)

Bits	Name	Description
		1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting state (**p_waiting** signal asserted) 010 Halted State (**p_halted** signal asserted) 011 Reserved 1XX Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

62.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related "window" for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z4251n3) Core Debug Support chapter.

PTS	PTE	DTS	DTE	0
31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0xB; DCR - 375; Read/Write; Reset - 0x0

Figure 62-9. Watchpoint Trigger (WT) Register

Table 62-16 details the Watchpoint Trigger register fields.

Table 62-16. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	PTE - Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 .

Table continues on the next page...

Table 62-16. WT field descriptions (continued)

Bits	Name	Description
		. 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

PTST																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x35; DCR - 412; Read/Write; Reset - 0x0

Figure 62-10. Program Trace Start Trigger Control (PTSTC) register

Table 62-17 details the PTSTC register fields.

Table 62-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11

Table 62-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTET																															

Nexus Reg# - 0x36; DCR - 413; Read/Write; Reset - 0x0

Figure 62-11. Program Trace End Trigger Control (PTETC) register

Table 62-18 details the PTETC register fields.

Table 62-18. PTETC field descriptions

Bits	Name	Description
31:0	PTET	PTET - Program Trace End Trigger Control
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2

Table 62-18. PTETC field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27 XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29 X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		DTST																													

Nexus Reg# - 0x37; DCR - 414; Read/Write; Reset - 0x0

Figure 62-12. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 62-19. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27 X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		DTET																													

Nexus Reg# - 038; DCR - 415; Read/Write; Reset - 0x0

Figure 62-13. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 62-20. DTET field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27

Table 62-20. DTET field descriptions

Bits	Name	Description
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

62.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages (DC1$_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

WEM
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x33; DCR - 411; Read/Write; Reset - 0x0

Figure 62-14. Watchpoint Mask Register

The following table details the Watchpoint Trigger register fields.

Table 62-21. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Watchpoint #0 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Watchpoint #1 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Watchpoint #2 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Watchpoint #3 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Watchpoint #4 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Watchpoint #5 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Watchpoint #6 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Watchpoint #7 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Watchpoint #8 enabled for WTM XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Watchpoint #9 enabled for WTM XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Watchpoint #10 enabled for WTM XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Watchpoint #11 enabled for WTM XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Watchpoint #12 enabled for WTM XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Watchpoint #13 enabled for WTM XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Watchpoint #14 enabled for WTM XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Watchpoint #15 enabled for WTM

Table 62-21. WMSK field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Watchpoint #16 enabled for WTM XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Watchpoint #17 enabled for WTM XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Watchpoint #18 enabled for WTM XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Watchpoint #19 enabled for WTM XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Watchpoint #20 enabled for WTM XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Watchpoint #21 enabled for WTM XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Watchpoint #22 enabled for WTM XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Watchpoint #23 enabled for WTM XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #24 enabled for WTM XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #25 enabled for WTM XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #26 enabled for WTM XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #27 enabled for WTM XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #28 enabled for WTM XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #29 enabled for WTM X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #30 enabled for WTM 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #31 enabled for WTM

62.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		SPTHOLD		0						SPEN						0		STTHOLD		0											STEN

Nexus Reg# - 0x32; DCR - 410; Read/Write; Reset - 0x0

Figure 62-15. Nexus Overrun Control Register

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 - Suppression threshold is when message queues are 1/4 full 01 - Suppression threshold is when message queues are 1/2 full 10 - Suppression threshold is when message queues are 3/4 full 11 - Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 - Suppression is disabled xxxxx1 - Ownership Trace message suppression is enabled xxxx1x - Data Trace message suppression is enabled xxx1xx - Program Trace message suppression is enabled xx1xxx - Watchpoint Trace message suppression is enabled x1xxxx - Reserved 1xxxxx - Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 - Stall threshold is when message queues are 1/4 full 01 - Stall threshold is when message queues are 1/2 full 10 - Stall threshold is when message queues are 3/4 full 11 - Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 - Stalling is disabled 1 - Stalling is enabled

Figure 62-16. Nexus Overrun Control Register Fields

62.4.10 Data Trace Control Register (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

RWT1		RWT2		RWT3		RWT4		0								STDC1	STDC2	STDC3	STDC4	0				RC1	RC2	RC3	RC4	DI	0		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xD; DCR - 376; Read/Write; Reset - 0x0

Figure 62-17. Data Trace Control Register

The following table details the Data Trace Control register fields.

Table 62-22. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range

Table continues on the next page...

Table 62-22. DTC field descriptions (continued)

Bits	Name	Description
		1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. The setting of the DI bit also affects the information provided on the data trace port outputs (See the "Data Trace Port Signals" section in the Core (e200z4251n3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

62.4.11 Data Trace Start Address Registers (DTSA1-4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0xE; DCR - 377; Read/Write; Reset - 0x0

Figure 62-18. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0xF; DCR - 378; Read/Write; Reset - 0x0

Figure 62-19. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x10; DCR - 379; Read/Write; Reset - 0x0

Figure 62-20. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x11; DCR - 380; Read/Write; Reset - 0x0

Figure 62-21. Data Trace Start Address 4 (DTSA4) register

62.4.12 Data Trace End Address (DTEA1-4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x12; DCR - 381; Read/Write; Reset - 0x0

Figure 62-22. Data Trace End Address 1 (DTEA1) register

Data Trace End Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x13; DCR - 382; Read/Write; Reset - 0x0

Figure 62-23. Data Trace End Address 2 (DTEA2) register

Data Trace End Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x14; DCR - 383; Read/Write; Reset - 0x0

Figure 62-24. Data Trace End Address 3 (DTEA3) register

Data Trace End Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x15; DCR - 408; Read/Write; Reset - 0x0

Figure 62-25. Data Trace End Address 4 (DTEA4) register

Table 62-23 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 62-23. Data trace - address range options

Programmed values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA -> <- DTEA
DTSA < DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace
DTSA = DTEA	N/A	Invalid Range - no trace

Note

> DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are *inclusive* of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are *exclusive* of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

62.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes.

Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values.

The RWCS Register also provides Read/Write Access Status information per Table 62-25.

AC	RW	SZ	MAP	PR	ATTR	0	CNT	ERR	DV
31	30	29 28 27	26 25 24	23 22	21 20 19 18	17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1	0

Nexus Reg# - 0x7; Read/Write[1] ; Reset - 0x0

Figure 62-26. Read/Write Access Control/Status (RWCS) register

NOTES:

[1] ERR and DV are read-only

Table 62-24. RWCS field description

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 100 - 111 Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map 001-111 Reserved
RWCS[23:22]	PR[1]	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	RES - Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR	Read/Write Access Error (see Table 62-25)
RWCS[0]	DV[2]	Read/Write Access Data Valid (see Table 62-25)

[1] The priority functionality is not currently implemented.

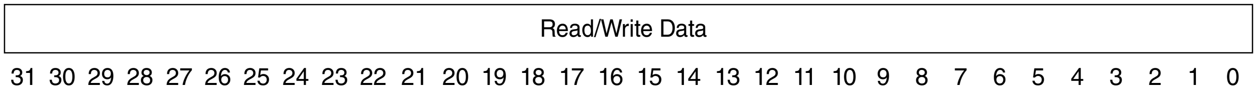
[2] ERR and DV are read-only.

Table 62-25. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

62.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.



Nexus Reg# - 0xA; Read/Write; Reset - 0x0

Figure 62-27. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. Table 62-26 shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 62-26. RWD data placement for ransfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	-	-	-	X
Half	x x 0	0 0 1	-	-	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword first RWD pass (low order data) second RWD pass (high order data)	0 0 0	0 1 1	 X X	 X X	 X X	 X X

Table Notes:

"X" indicates byte lanes with valid data

"-" indicates byte lanes that will contain unused data.

Table 62-27 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 62-27. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	-	-	-	AHB[7:0]
Byte @001	0 0 1	-	-	-	AHB[15:8]
Byte @010	0 1 0	-	-	-	AHB[23:16]
Byte @011	0 1 1	-	-	-	AHB[31:24]
Byte @100	1 0 0	-	-	-	AHB[39:32]
Byte @101	1 0 1	-	-	-	AHB[47:40]
Byte @110	1 1 0	-	-	-	AHB[55:48]
Byte @111	1 1 1	-	-	-	AHB[63:56]
Half @000	0 0 0	-	-	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	-	-	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	-	-	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	-	-	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

"-" indicates byte lanes that will contain unused data.

62.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Read/Write Address

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x9; Read/Write; Reset - 0x0

Figure 62-28. Read/Write Access Address (RWA) register

62.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in Table 62-8.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See IEEE 1149.1 (JTAG) RD/WR sequences.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see Table 62-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

(7bits)	(1 bit)
Nexus Register Index	R/W

RESET Value: 0x0

Nexus Register Index	Selected from values in Table 62-8
Read/Write (R/W):	0 - Read 1 - Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.

   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the "Capture-DR" state.

   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the "Update-DR" state.

62.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. Table 62-8 shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

62.7 Nexus message fields

Nexus messages are composed of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

62.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

62.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

62.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

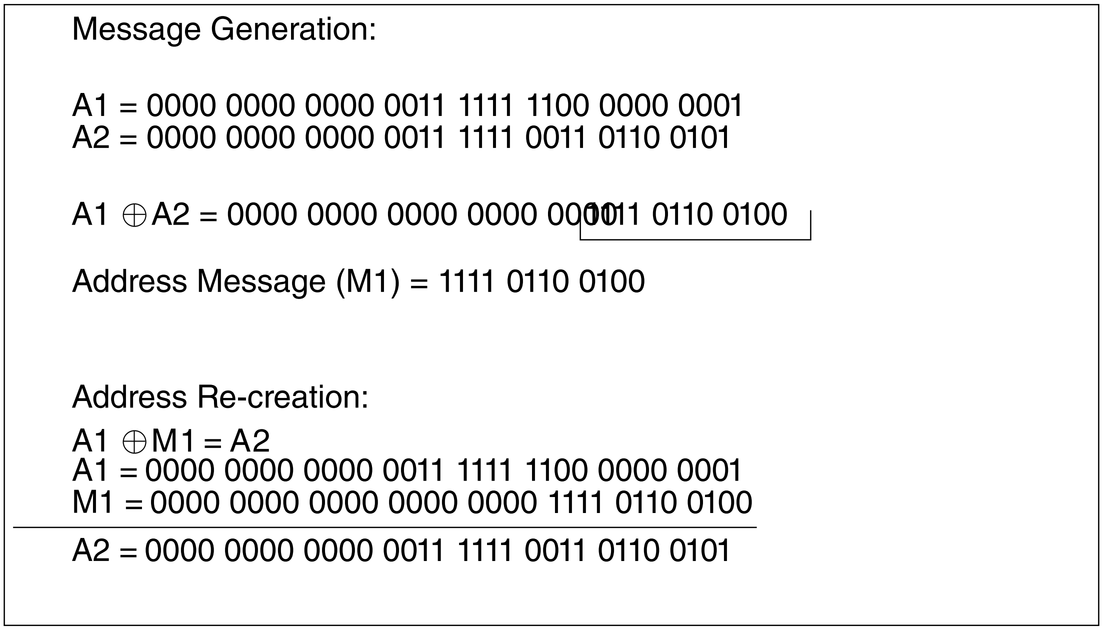
The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365



Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 1111 0110 0100

Address Message (M1) = 1111 0110 0100

Address Re-creation:

A1 ⊕ M1 = A2
A1 = 0000 0000 0000 0011 1111 1100 0000 0001
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

Figure 62-29. Relative address generation and recreation

62.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

62.7.5 Timestamp field (TSTAMP)

Timestamps may be optionally applied to messages. Enabling of timestamping is controlled by the nex_tsen input signal. The timestamping mode (timestamp frequency of occurrence) is controlled by the value of nex_tsmode[0:1] and the timestamp value is based on the nex_timebase[0:29] inputs. These signals are first synchronized to the nex_clk clock domain, and for nex_timebase[0:29], a gray-code to binary value conversion is performed. The timestamp value is applied to messages as they exit the message queues. Leading zeros in the timestamp value are removed. The table shows the nex_tsmode[0:1] encodings used for the timestamp mode.

Table 62-28. nex_tsmode[0:1] timestamp mode encoding

nex_tsmode[0:1]	Timestamp Mode
00	Timestamp applied to every message
01	Timestamp applied to every 4th message
10	Timestamp applied to every 16th message
11	Timestamp applied to every 64th message

62.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

62.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

62.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to Nexus Overrun Control Register (OVCR) for complete programming options.

62.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in

Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

62.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

Table 62-29 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in Table 62-29.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM ->DQM -> $\mathrm{PCM}_{\mathrm{PIDMSG}}$ -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 62-29. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A[1]
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM - PID/NPIDR update (Program Correlation Message)	2	OTM Error Message

Table continues on the next page...

Table 62-29. Message Type Priority and Message Dropped Responses (continued)

Message Type	Message	Priority	Message Dropped Response
OT (Ownership)	OTM - PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message[2]
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

62.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

62.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

62.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

62.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

62.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

62.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

(8 bits)	(4 bits)	(6 bits)
DS[31:24]	Src. Proc.	TCODE (000000)

Fixed length = 18 bits

Figure 62-30. Debug Status message format

62.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note

> The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see Table 62-3 and Table 62-4):

(6 bits)	(4 bits)	(4 bits)	(6 bits)
Error Code	Error Type	Src. Proc.	TCODE (001000)

Fixed length = 20 bits

Figure 62-31. Error message format

62.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

62.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

62.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z4251n3) processors contain a *Power ISA 2.06* defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note

> The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

Nexus 3 Process ID																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 517; Read/Write; Reset: - 0

Figure 62-32. Nexus 3 Process ID Register (NPIDR)

Note

> OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for

PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.

- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting DC1$_{\text{POTD}}$.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15-46 bits

Figure 62-33. Ownership Trace Message format

62.12 Program trace

This section details the program trace mechanism supported by Nexus3 for the e200z4251n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z4251n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

62.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Certain instructions (**zvselh**) generate a pair of predicate bits that are both reported as consecutive bits in the history field. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

62.12.1.1 Indirect Branch Message instructions

Table 62-30 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 62-30. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	**Indirect branch with Sync, target address is initial instruction, count = 1**

62.12.1.2 Direct Branch Message Instructions

Table 62-31 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 62-31. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b, ba, bl, bla, bc, bca, bcl, bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

62.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken. Certain instructions (**zvselh**) generate a pair of predicate bits that are both reported as consecutive bits in the history field.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

62.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

62.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

62.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

Figure 62-34. Indirect branch message (history) format

62.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

Figure 62-35. Indirect branch message format

62.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

(1-8 bits)	(4 bits)	(6 bits)
Sequence Count	Src. Proc.	TCODE (000011)

Max length = 18 bits; Min length = 11bits

Figure 62-36. Direct Branch message format

Note

When DC1_{PTM} is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

62.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

62.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

62.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false. For the **zvselh** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 62-32. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**[1]
zvselh instruction	00,01,10, or 11	**zvselh**

[1] If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.1

62.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

62.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/ predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow, or, if program trace is in history mode, since the last instruction that recorded history information.

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
RDATA	RCODE	Src. Proc.	TCODE (011011)

Max length = 46 bits; Min length = 15 bits

Figure 62-37. Resource Full Message format

The following table shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 62-33. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

62.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a "Branch and Link" instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}='0' and $\text{DC4}_{\text{PTMARK}}$='1'.
- When a write to the process ID register selected by DC1_{OTS} (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to Table 62-6 for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

(1-32 bits)	(1-8 bits)	(2 bits)	(4 bits)	(4 bits)	(6 bits)
Branch History	Sequence Count	CDF*	EVCOD	Src. Proc.	TCODE (100001)

Max length = 56 bits; Min length = 18 bits

* - CDF=01,
EVCODE = Any but 0101, 110

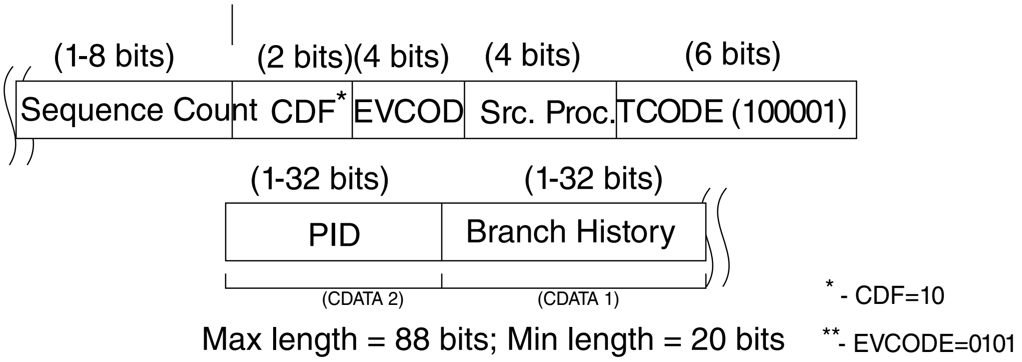


Figure 62-38. Program Correlation message formats

62.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

62.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

62.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 62-34):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to MSR_{PMM} → '1' with $\text{DC4}_{\text{PTMARK}}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Exception conditions that result in Program Trace Synchronization are summarized in Table 62-34.

Table 62-34. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.

Table continues on the next page...

Table 62-34. Program Trace exception summary (continued)

Exception Condition	Exception Handling
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

62.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 62-34):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

Figure 62-39. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. messages is as follows:

(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Figure 62-40. Indirect Branch History w/ Sync. message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in Table 62-35.

Table 62-35. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> $\text{PCM}_{\text{PIDMSG}}$ → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

62.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $\text{DC4}_{\text{PTMARK}}$

62.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

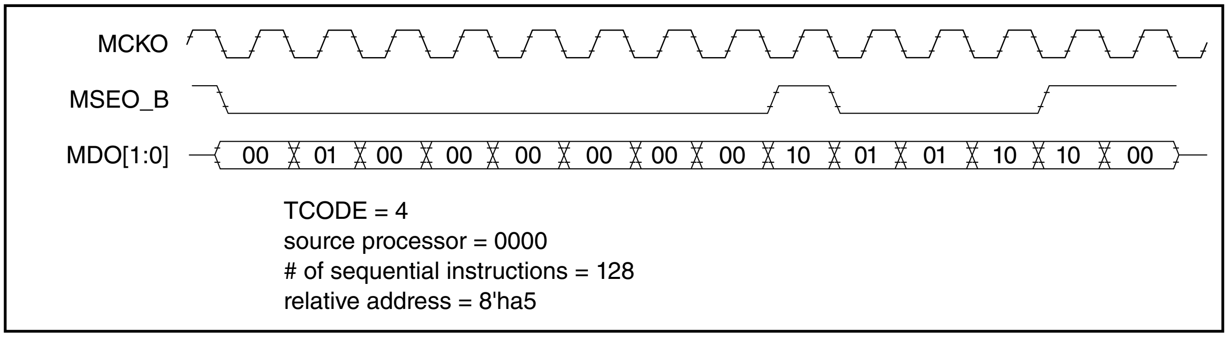


Figure 62-41. Program Trace - Indirect Branch message (Traditional)

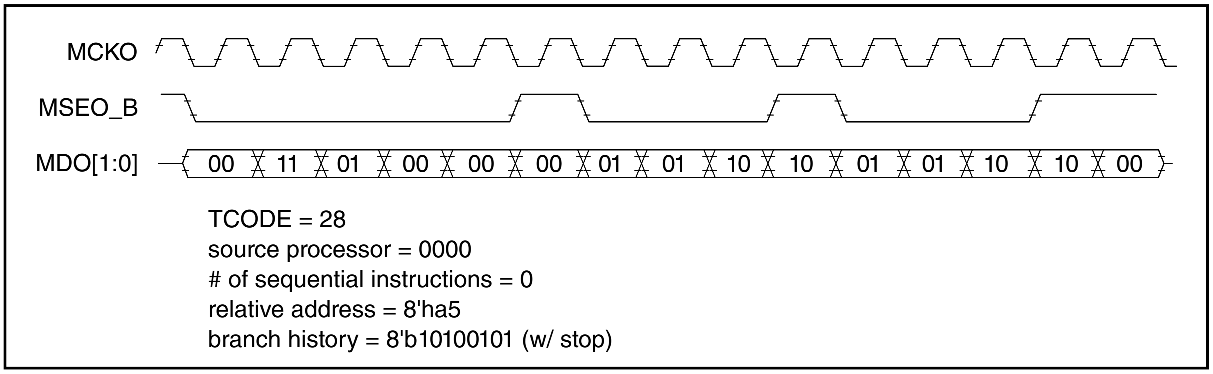


Figure 62-42. Program Trace - Indirect Branch message (history)

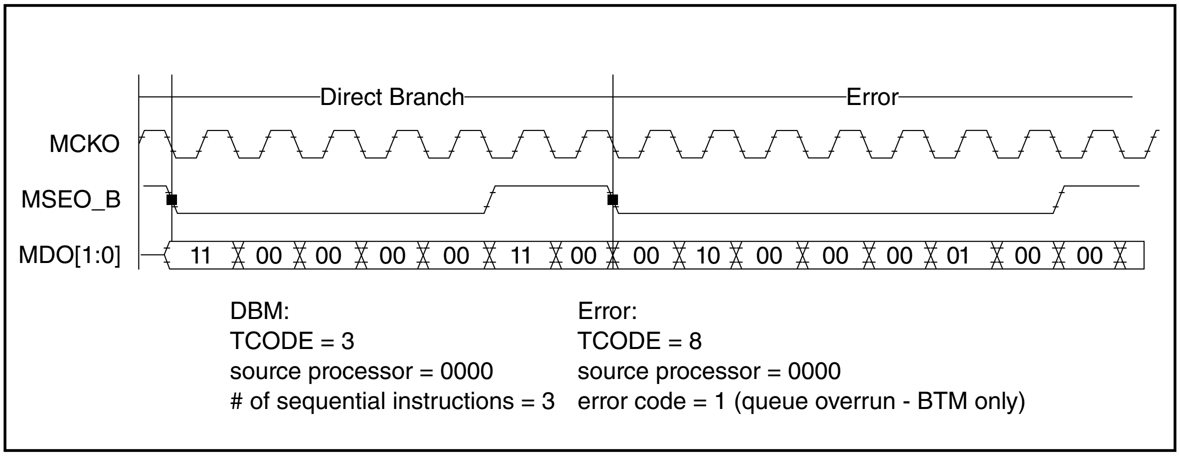


Figure 62-43. Program Trace - Direct Branch (traditional) & Error messages

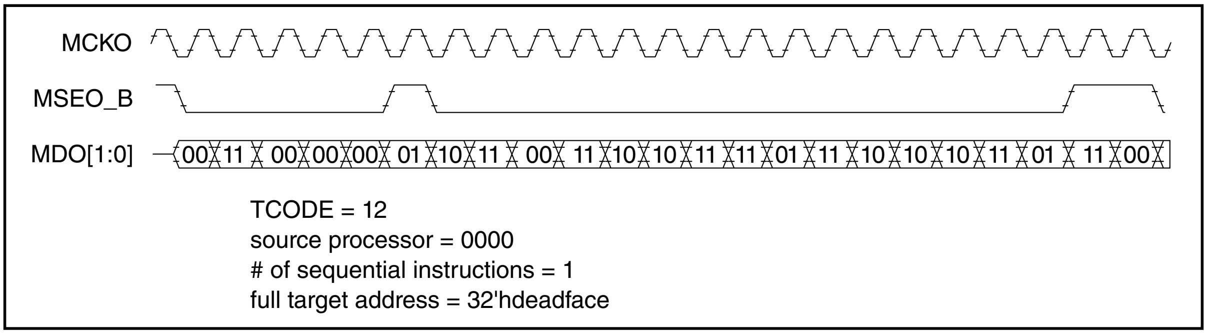


Figure 62-44. Program Trace - Indirect Branch w/ Sync. message

62.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

62.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note

Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z4251n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of the following ways:

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).

62.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

62.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 62-45. Data Write message format

62.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 62-46. Data Read fessage format

Note

* Core (e200z4251n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Core (e200z4251n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

62.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see Table 62-36):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

Figure 62-47. Data Write/Read with Sync. message format

Exception conditions that result in Data Trace Synchronization are summarized in Table 62-36.

Table 62-36. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → $\text{PCM}_{\text{PIDMSG}}$ → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

62.13.3 DTM Operation

62.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

62.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

62.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

62.13.3.4 Bus Cycle special cases

Table 62-37. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$)	Data Trace Message discarded
Bus cycle completed without error[1]	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

[1] Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

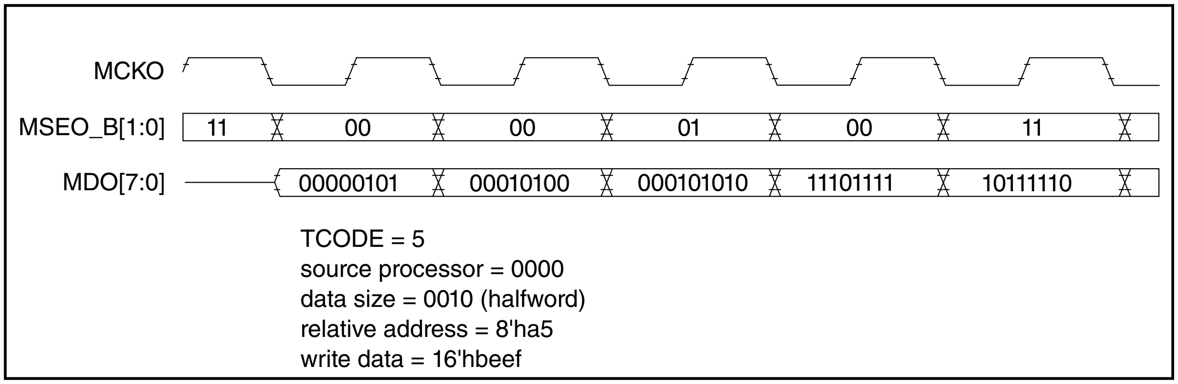
Note

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0, or unless the misaligned access wraps over the end of a circular buffer when using the LSP specialized load or store with circular addressing. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z4251n3) Core Debug Support chapter.)

Note

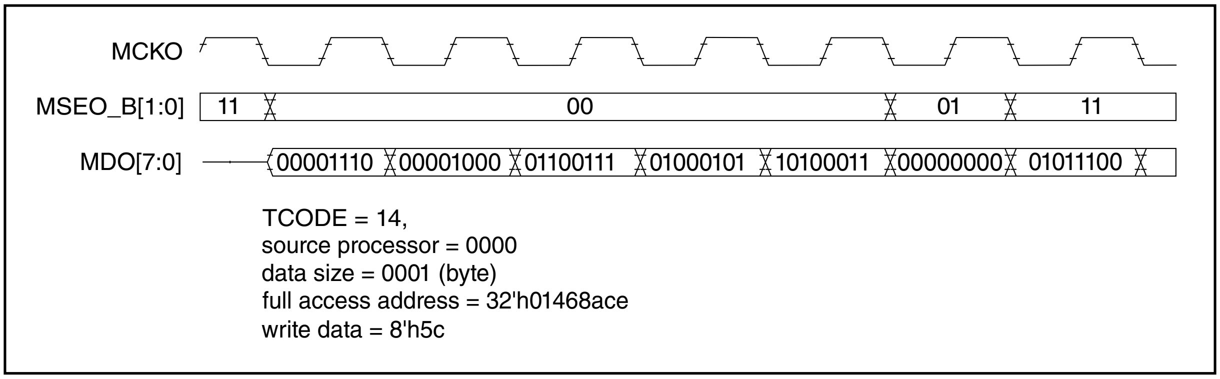
A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

62.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)



TCODE = 5
source processor = 0000
data size = 0010 (halfword)
relative address = 8'ha5
write data = 16'hbeef

Figure 62-48. Data Trace - Data Write Message



TCODE = 14,
source processor = 0000
data size = 0001 (byte)
full access address = 32'h01468ace
write data = 8'h5c

Figure 62-49. Data Trace — Data Read w/ Sync message

62.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

62.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $\text{DEVENT}_{\text{DQTAG}}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

62.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

62.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the "External Debug Resource Allocation Control Register (EDBRAC0)" section in the Core (e200z4251n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

(1-32 bits)	(8 bits)	(4 bits)	(6 bits)
DQDATA	DQTAG	Src. Proc.	TCODE (000111)

Max length = 50 bits; Min length = 19 bits

Figure 62-50. Data Acquisition message format

62.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z4251n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the WMSK_{WEM} field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See Table 62-40 for details on **evto_b**.

Watchpoint information is messaged out in the following format:

(1-30 bits)	(4 bits)	(6 bits)
Watchpoint Source	Src. Proc.	TCODE (001111)

Variable length = 11-40 bits

Figure 62-51. Watchpoint message format

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 62-38. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
00000000000000000000000000000 - No Watchpoints enabled for Watchpoint Trace Messaging	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 - Watchpoint #0 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X - Watchpoint #1 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX - Watchpoint #2 enabled for WTM	

Table 62-38. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX - Watchpoint #3 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX - Watchpoint #4 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX - Watchpoint #5 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX - Watchpoint #6 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX - Watchpoint #7 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX - Watchpoint #8 enabled for WTM	
XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX - Watchpoint #9 enabled for WTM	
XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX - Watchpoint #10 enabled for WTM	
XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX - Watchpoint #11 enabled for WTM	
XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX - Watchpoint #12 enabled for WTM	
XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX - Watchpoint #13 enabled for WTM	
XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX - Watchpoint #14 enabled for WTM	
XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX - Watchpoint #15 enabled for WTM	
XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX - Watchpoint #16 enabled for WTM	
XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX - Watchpoint #17 enabled for WTM	
XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX - Watchpoint #18 enabled for WTM	
XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX - Watchpoint #19 enabled for WTM	
XXXXXXXXX1XXXXXXXXXXXXXXXXXXXX - Watchpoint #20 enabled for WTM	
XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX - Watchpoint #21 enabled for WTM	
XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX - Watchpoint #22 enabled for WTM	
XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #23 enabled for WTM	
XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #24 enabled for WTM	
XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #25 enabled for WTM	
XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #26 enabled for WTM	
XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #27 enabled for WTM	
X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #28 enabled for WTM	
1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #29 enabled for WTM	

62.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

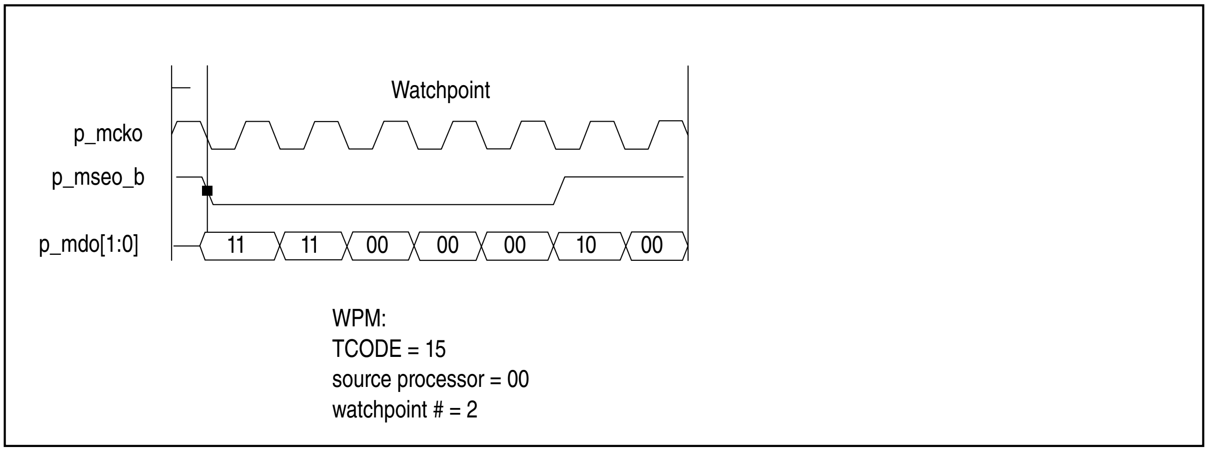


Figure 62-52. Watchpoint Message & Watchpoint Error message

62.16 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/ OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The $\mathrm{RWCS}_{\mathrm{ATTR}}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note

Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

62.16.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'1 (write access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

   Note

   Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data (RWD) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD register to the memory mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see Table 62-40 for details) and clears the ERR bit in the RWCS register. Otherwise, if the access completes with an error, the ERR bit will be set to indicate an error has

occurred, and the **nex_rdy_b** pin will not be asserted. Once the DV bit has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note

Only the **nex_ahb_start**, and **nex_rdy_b** pins, as well as the AC, DV, and ERR bits within the RWCS, provide Read/Write Access status to the external development tool.

62.16.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block then arbitrates for the AHB system bus and transfers the first data value from the RWD register to the memory-mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output is asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA register is incremented to the next word size (specified in the SZ field) andthe number from the CNT field is decremented. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS register. Otherwise, the ERR bit is set and the DV bit is cleared to indicate an error has occurred, , the block transfer is aborted, and the AC bit in the RWCS register is cleared. Clearing the DV bit indicates that the device is ready for the next access in the block transfer, or that an error has occurred.
3. If the AC bit has not been cleared due to an error, repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block write access.

Note

The actual RWA register value as well as the CNT field within the RWCS register are not changed when executing a block write access. The original values can be read by the external development tool at any time.

62.16.3 Single read access

1. Initialize the Read/Write Access Address (RWA)register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'0 (read access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

   Note

   An Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 62-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred. The **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit is cleared in either case.
4. The data can then be read from the Read/Write Access Data register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

Note

Only the **nex_ahb_start**and **nex_rdy_b** pins as well as the AC, DV, and ERR bits within the RWCS register provide Read/Write Access status to the external development tool.

62.16.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 62-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS register, and asserts the **nex_rdy_b** pin (see Table 62-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the RWD register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block read access.

Note

The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note

The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

62.16.5 Error handling

The Nexus 3 module handles various error conditions as follows:

62.16.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set
3. The Error Message is sent (TCODE = 8) indicating Read/Write error

62.16.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.

a. If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.

b. If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

62.16.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

(5 bits)	(4 bits)	(6 bits)
Error Code (00011)	Src. Proc.	TCODE (001000)

Fixed length = 15 bits

Figure 62-53. Error message format1

62.17 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

62.17.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **evto_b** pin, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in Table 62-40. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see Nexus 3 Register Access via JTAG/OnCE for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the "JTAG/OnCE Pins" section of the Core (e200z4251n3) Core Debug Support chapter.

Table 62-39. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the "Shift-IR" and "Shift-DR" controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in Table 62-40.

Table 62-40. Nexus 3 Auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation.
nex_ahb_start	O	AHB Start (**nex_ahb_start**) is an output pin used to indicate to the external tool or SoC that the Nexus block is requesting the next Read/Write Access on the system bus. If Nexus is enabled, this signal is asserted upon an acknowledged request to start an AHB system bus transfer (Nexus read or write) and is pulsed asserted for one **nex_clk** clock period, corresponding to the first clock of the address phase of the transfer. Upon exit from system reset or if Nexus is disabled, **nex_ahb_start** remains de-asserted.
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted.
evto_b	O	Event Out (**evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1. One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00

Table continues on the next page...

Table 62-40. Nexus 3 Auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
		2. Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1. Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2. Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_ext_src_id[0:3]	I	**nex_ext_src_id[0:3]** is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time.
nex_sfwcntl_en]	I	**nex_sfwcntl_en** is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to Auxiliary port arbitration for detail on Nexus port arbitration.

Table 62-41. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 62-43.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant **(nar_aux_grant)** is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

62.17.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in Table 62-42. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the MPC5744P.

Table 62-42. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 62-54 illustrates the state diagram for single pin MSEO transfers.

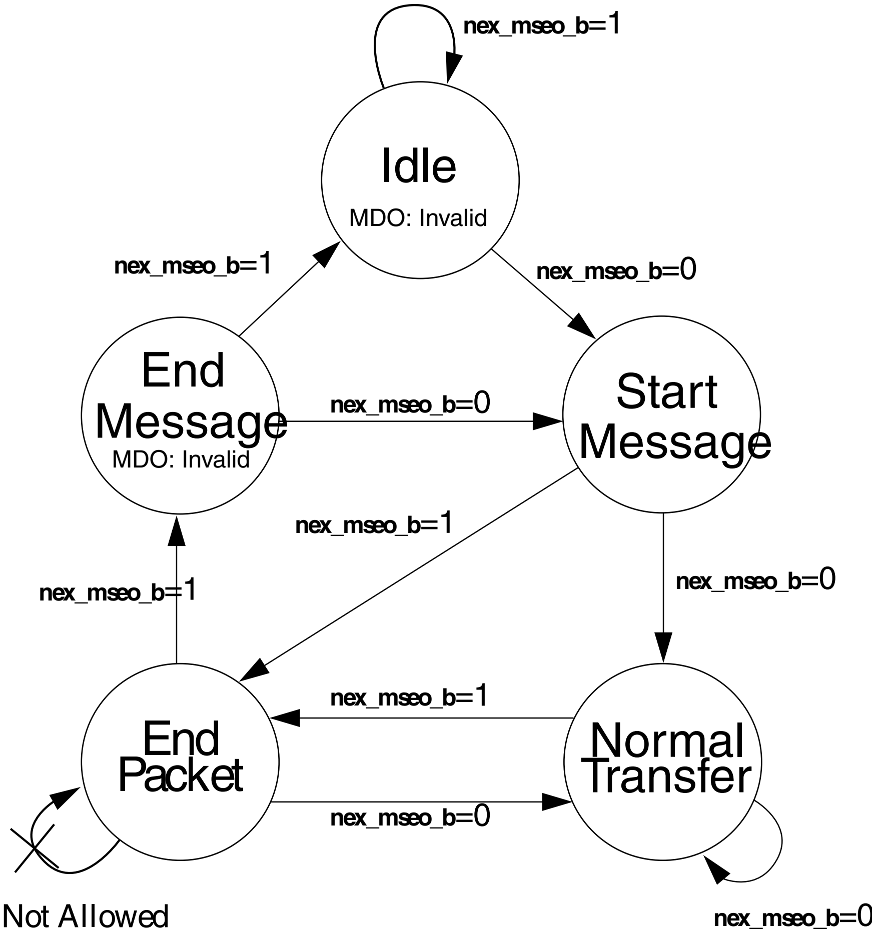


Figure 62-54. Single pin MSEO transfers

Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 62-55 illustrates the state diagram for dual pin MSEO transfers.

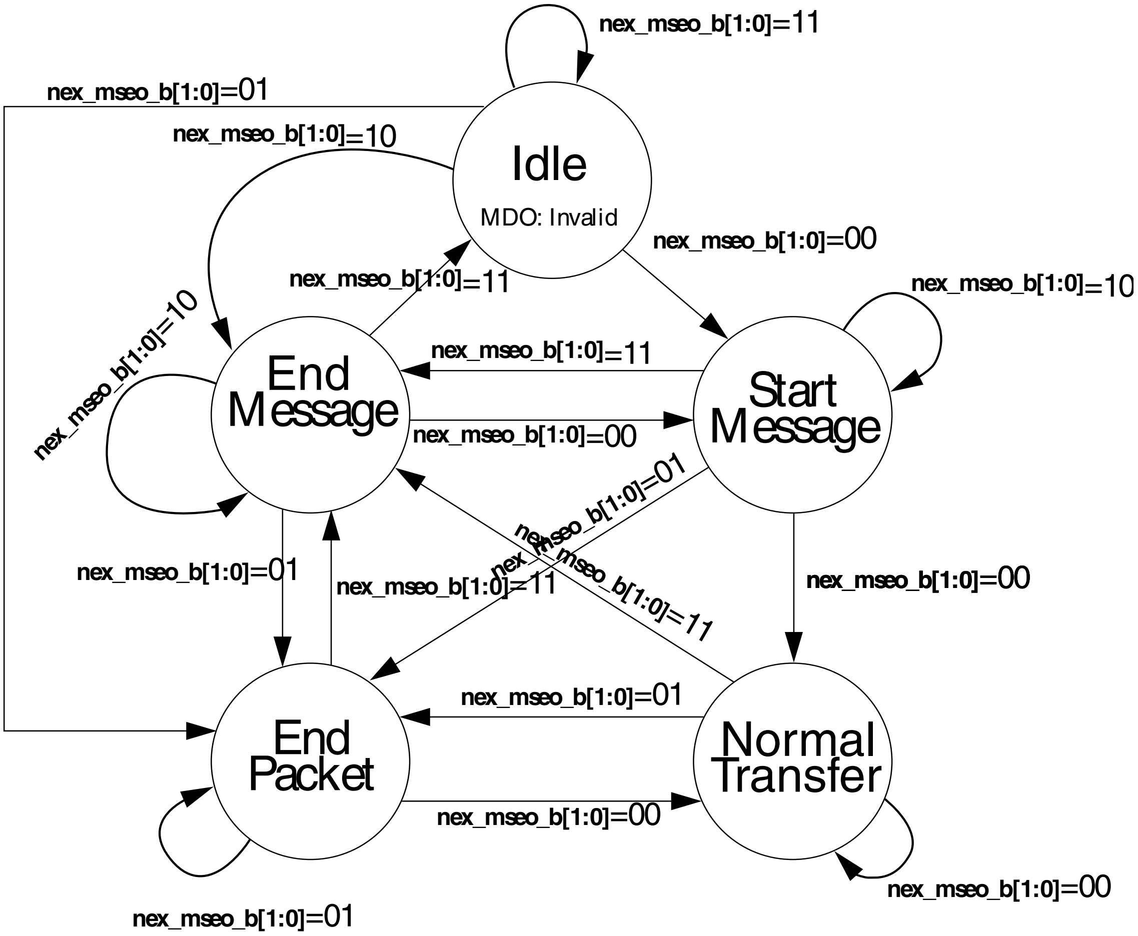


Figure 62-55. Dual pin MSEO transfers

The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive "End Packet" states. This can be an advantage when small, variable sized packets are transferred.

Note

> The "End Message" state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

62.18 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

62.19 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in Table 62-43 below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in Pin protocol. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 62-43. MDO Request Encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

62.20 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 62-44 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. Table 62-45 illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 62-44. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 62-45. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A P	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 62-46 and Table 62-47 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 62-46. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 62-47. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 62-48 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and Table 62-49 illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 62-48. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 62-49. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

62.21 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the MPC5744P Data Sheet.

62.22 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

62.22.1 JTAG sequence for accessing internal Nexus registers

Table 62-50. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

62.22.2 JTAG sequence for read access of memory-mapped resources

Table 62-51. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

62.22.3 JTAG sequence for write access of memory-mapped resources

Table 62-52. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** pin
8	—	If CNT > 0, go back to Step #5

Chapter 63
Nexus Crossbar Multi-Master Client (NXMC)

63.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2012 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools.

63.1.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources.
- Multi-master tracing capability for multiple master accesses
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on internal triggers.

- Nexus Auxiliary port for higher data throughput
  - 4 MDO (Message Data Out) pins
  - Two $\overline{\text{MSEO}}$ (Message Start/End Out) pins
  - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
  - One $\overline{\text{EVTI}}$ (Event In) pin
  - One MCKO (Message Clock Out) pin
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

63.2 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NPC. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$
- Two $\overline{\text{MSEO}}$
- 4 MDOs
- One MCKO

These pins are described in Table 63-2.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see IEEE 1149.1 (JTAG) Test Access Port for details). A separate JTAG TAP controller is instantiated within the NXMC. The following table describes the JTAG pins.

Table 63-1. JTAG pins for NXMC

JTAG port	Input/ output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 63-2. NXMC auxiliary pins

Auxiliary port	Input/ output	Description
MCKO	O	Message Clock Out (MCKO) is a Nexus generated output clock to development tools for timing of MDO and $\overline{\text{MSEO}}$ pin functions. MCKO is programmable through the DC Register.
MDO[4:0]	O	Message Data Out (MDO[4:0]) are output pins used for DTM and WPM (watch point message). External sampling of MDO by the development tool shall occur on rising edge of the Nexus clock (MCKO).
$\overline{\text{MSEO}}$[1:0]	O	Message Start/End Out ($\overline{\text{MSEO}}$) are output pins which indicate when a message on the MDO pins has started, when a variable length packet has ended, and when the message has ended. External sampling of $\overline{\text{MSEO}}$ shall occur on rising edge of the Nexus clock (MCKO).
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: • one of the internal or external watchpoints has occurred and EOC = 2'b00 • system-level debug mode was entered (ipg_debug) and EOC = 2'b01 $\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (see Enabling NXMC operation), assertion initiates data trace synchronization messages (provided data trace is enabled).

63.3 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in the following table. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 63-3. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 63-4)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)

Table continues on the next page...

Table 63-3. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 63-4)
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (See Table 63-5)
	14	14	ECODE	fixed	Error code (See Table 63-6)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 63-4)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 63-4)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (See Table 63-22)

Table 63-4. Data trace size (DSZ) encodings (TCODE = 58, 59, 60, 61)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 63-5. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 63-6. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description[1]
13 to 8	7 to 0	
Source (SRC)		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	Reserved
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

63.4 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See IEEE 1149.1 (JTAG) Test Access Port for details. The complete list of registers is shown in this table.

Table 63-7. NXMC registers

Nexus register	Nexus access opcode (hex)	Read/ write	Read address (hex)	Write address (hex)
Development Control 1 (DC1)	2	R/W	04	05
Development Control 2 (DC2)	3	R/W	06	07
Watchpoint Trigger (WT)	B	R/W	16	17
Data Trace Control (DTC)	D	R/W	1A	1B
Data Trace Start Address1 (DTSA1)	E	R/W	1C	1D
Data Trace Start Address2 (DTSA2)	F	R/W	1E	1F
Data Trace End Address1 (DTEA1)	12	R/W	24	25
Data Trace End Address2 (DTEA2)	13	R/W	26	27
Breakpoint/Watchpoint Control Register1 (BWC1)	16	R/W	2C	2D
Breakpoint/Watchpoint Control Register2 (BWC2)	17	R/W	2E	2F
Breakpoint/Watchpoint Address Register1 (BWA1)	1E	R/W	3C	3D
Breakpoint/Watchpoint Address Register2 (BWA2)	1F	R/W	3E	3F

63.4.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	EOC		0	0	WEN	0	0	0	0	0	0	0	0	0	C	0	0	0	0	0	0	0	0	0	EIC		TM		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC1 is described in this table.

Table 63-8. DC1 register field description

Field	Description
31:29	Reserved for future functionality (read as 0)
28:27 EOC	$\overline{\text{EVTO}}$ Control. 00 $\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal) 01 $\overline{\text{EVTO}}$ upon entry into system-level Debug Mode 1x Reserved
26:25	Reserved for future functionality (read as 0)
24 WEN	Watchpoint Trace Enable. 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
23:5	Reserved for future functionality (read as 0)
4:3 EIC	$\overline{\text{EVTI}}$ Control. 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled 11 Reserved
2:0 TM	Trace Mode. 000 No Trace 1xx Reserved x1x Data Trace enabled xx1 Reserved

63.4.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC2.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC2 is described in this table.

Table 63-9. DC2 register field description

Field	Description	
31:24 EWC	$\overline{\text{EVTO}}$ Watchpoint Configuration[1].	
	00000000	No Watchpoints trigger $\overline{\text{EVTO}}$
	1XXXXXXX	Reserved
	X1XXXXXX	Reserved
	XX1XXXXX	Reserved
	XXX1XXXX	Reserved
	XXXX1XXX	Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$
	XXXXX1XX	Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$
	XXXXXX1X	Reserved
	XXXXXXX1	Reserved
23:0	Reserved for future functionality (read as 0)	

[1] The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.

63.4.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable.

The WT bits can be used to produce an address related window for triggering trace messages. This table shows the format of the WT register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DTS			DTE			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The WT register is described in this table.

Table 63-10. WT register field description

Field	Description[1]	
31:26	Reserved for future functionality (read as 0)	
25:23 DTS	Data Trace Start Control.	
	000	Trigger disabled
	001	Reserved
	010	Reserved
	011	Reserved

Table continues on the next page...

Table 63-10. WT register field description (continued)

Field	Description[1]
	100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
22:20 DTE	Data Trace End Control. 000 Trigger disabled 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
19:0	Reserved for future functionality (read as 0)

[1] The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

63.4.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXMC module. This table shows the format of the DTC register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RWT1		RWT2		0	0	0	MME	0	0			MID				0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DTC register is described in this table.

Table 63-11. DTC field description

Field	Description
31:30 RWT1	Read/Write Trace 1. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace

Table continues on the next page...

Table 63-11. DTC field description (continued)

Field	Description
29:28 RWT2	Read/Write Trace 2. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
27:25	Reserved for future functionality
24 MME	Multi Master Tracing enable. 0 Tracing enabled only for Master with ID = MID (DTC[19:16]) 1 Tracing enabled for all the Masters
23:20	Reserved for future functionality
19:16 MID	ID of the system bus Master to be traced[1]
15:8	Reserved for future functionality
7 RC1	Range Control 1. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
6 RC2	Range Control 2. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
5:0	Reserved for future functionality

[1] The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

63.4.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. This table shows the format of the DTSA1 and DTSA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA TRACE START ADDRESS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

63.4.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. This table shows the format of the DTEA1 and DTEA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA TRACE END ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table illustrates the range that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 63-12. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range—no trace

Note

> DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.
>
> When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

63.4.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. This table shows the format of the BWC1 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1		BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC1 register is described in this table.

Table 63-13. BWC1 field description

Field	Description
31:30 BWE1	Breakpoint/Watchpoint #1 Enable. 00 Internal Nexus Watchpoint #1 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #1 enabled
29:28 BRW1	Breakpoint/Watchpoint #1 Read/Write Select. 00 Watchpoint #1 hit on read accesses 01 Watchpoint #1 hit on write accesses 10 Watchpoint #1 on read or write accesses 11 Reserved
27:18	Reserved for future functionality (read as 0)
17:16 BWR1	Breakpoint/Watchpoint #1 Register Compare. 00 No Register Compare (same as BWC1[31:30] = 2'b00) 01 Reserved 10 Compare with BWA1 value 11 Reserved
15 BWT1	Breakpoint/Watchpoint #1 Type. 0 Reserved 1 Watchpoint #1 on data accesses
14:0	Reserved for future functionality (read as 0)

63.4.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. This table shows the format of the BWC2 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2		BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC2 register is described in this table.

Table 63-14. BWC2 field description

Field	Description
31:30 BWE2	Breakpoint/Watchpoint #2 Enable. 00 Internal Nexus Watchpoint #2 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #2 enabled
29:28 BRW2	Breakpoint/Watchpoint #2 Read/Write Select. 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
27:18	Reserved for future functionality (read as 0)
17:16 BWR2	Breakpoint/Watchpoint #2 Register Compare. 00 No Register Compare (same as BWC2[31:30] = 2b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved
15 BWT2	Breakpoint/Watchpoint #2 Type. 0 Reserved 1 Watchpoint #2 on data accesses
14:0	Reserved for future functionality (read as 0)

63.4.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. This table shows the format of the BWA1 and BWA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BREAKPOINT / WATCHPOINT ADDRESS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

63.4.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in Table 63-7. For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

63.5 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of $\overline{\text{TRST}}$ and system reset de-assertion. The JTAG ID register should be read by the tool after negation of $\overline{\text{TRST}}$ and should be programmed before deassertion of system reset.

63.5.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in NXMC register access via JTAG.

63.5.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

63.5.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set, etc.

63.5.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR).

Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table 63-7.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table 63-7), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown below.

	7	6	5	4	3	2	1	0
R W	Nexus Register Index							R/W
Reset	0	0	0	0	0	0	0	0

Table 63-15. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in Table 63-7
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.

a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.

b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

63.6 Functional description

The following sections describe the functionality of the NXMC module.

63.6.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2012 standard.

63.6.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

63.6.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

63.6.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32 , 64. Data write message and data read message information is messaged out in the format shown in the following figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 63-16. DTM message format for DATA width = 64 bits

8/16/32/64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010 or 111011)

63.6.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

63.6.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in this figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 63-17. DTM error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

NOTE

The internal queue FIFO is very small. This causes FIFO overflows whenever Trace is enabled on any of the AHB clients and the AHB master module generates >4 burst accesses. If the customer is experiencing a lot of overflow error messages the amount of data snooped by the NXMC should be reduced.

63.6.1.2.4 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see Table 63-19):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message.
- Upon returning from a low power state, the first data trace message is a synchronization message.
- Upon returning from debug mode, the first data trace message is a synchronization message.
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message.
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred.
- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature.
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message.This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic.

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in the following figure. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 63-18. Data write/read with synchronization message format

8-64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100 or 111101)

Exception conditions that result in data trace synchronization are summarized in this table.

Table 63-19. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).
Collision Priority	All messages have the following priority: Error > WPM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

63.6.1.3 Enabling data trace messaging operation

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).

- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. (The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (i.e. DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure correct DTM functionality.

63.6.1.4 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

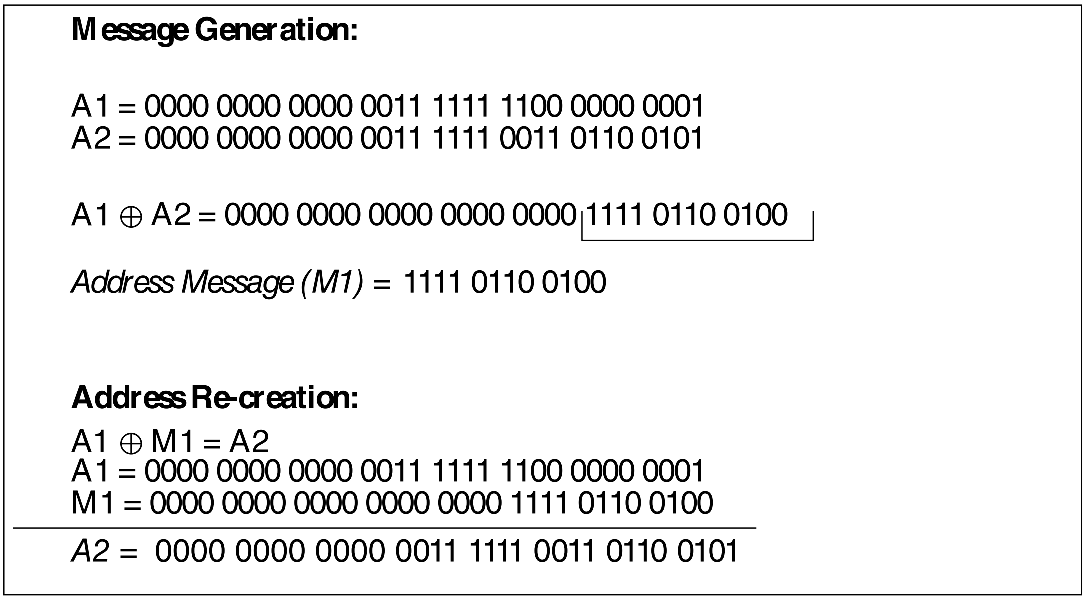
Note

If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > DTM).

63.6.1.5 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2012 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address (A1) = 0x0003FC01, the new address (A2) = 0x0003F365 as shown in this figure.



Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

$A1 \oplus A2 = 0000\ 0000\ 0000\ 0000\ 0000\ \underbrace{1111\ 0110\ 0100}$

Address Message (M1) = 1111 0110 0100

Address Re-creation:

$A1 \oplus M1 = A2$
A1 = 0000 0000 0000 0011 1111 1100 0000 0001
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

Figure 63-1. Relative addressing example

63.6.1.6 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

63.6.1.7 System bus cycle special cases

The system bus cycle special cases are shown in this table.

Table 63-20. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

63.6.1.8 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (see figure in Data write and data read messages for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

63.6.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2012. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

63.6.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

> In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (i.e. BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints—Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in System bus multi-master tracing).
- External watchpoints—The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level. The watchpoints are described in the device-specific chapter that describes how the modules are configured. If either of these signals asserts (i.e, on 0-to-1 transition), a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in Table 63-21 and Table 63-22.

Table 63-21. Watchpoint message format (fixed length = 18 bits)

8 bits	4 bits	6 bits
WPHIT (RRRRXXXX)	SRC	TCODE (001111)

Table 63-22. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1[1]	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

[1] R = reserved for future use (insert 0s).

63.6.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 0000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (0000000000011). Error information is messaged out in the format shown in this figure.

Table 63-23. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

Chapter 64
Nexus Port Controller (NPC)

64.1 Introduction

The following figure is a block diagram of the Nexus Port Controller (NPC) block.

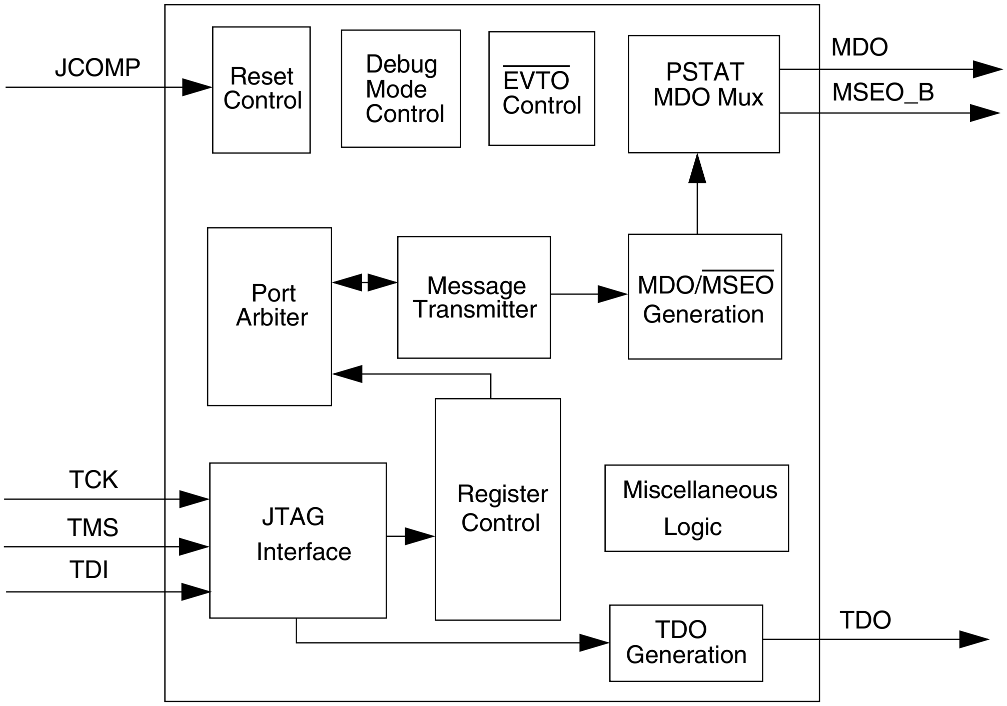


Figure 64-1. Nexus Port Controller Block Diagram

64.1.1 Overview

On a system-on-a-chip device, there are often multiple blocks that require development support. Each of these blocks implements a development interface based on the IEEE-ISTO 5001-2001 standard. The blocks share input and output ports that interface with the development tool. The NPC controls the usage of the input and output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

64.1.2 Features

The NPC block performs the following functions:

- Controls arbitration for ownership of the Nexus Output Port
- Nexus Device Identification Register and Messaging
- Generates MCKO enable and frequency division control signals
- Controls sharing of $\overline{\text{EVTO}}$
- Generates an MCKO clock gating control signal to enable gating of MCKO when the auxiliary output port is idle
- Control of the device-wide debug mode
- Generates asynchronous reset signal for Nexus blocks based on JCOMP input, and power-on reset status
- System clock status indication via MDO[0] following power-on reset
- Controls arbitration for ownership of the Nexus Aurora Link (NAL) in Aurora Link mode

64.1.3 Modes of operation

The NPC block uses the JCOMP input, and an internal power-on reset indication as its primary reset signals. Upon exit of reset, the mode of operation is determined by the Port Configuration Register (PCR) settings.

64.1.3.1 Reset

The NPC block is asynchronously placed in reset when either power-on reset is asserted, JCOMP is not set for Nexus access or the TAP controller state machine is in the Test-Logic-Reset state. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state.

Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. The NPC is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state
- The output port pins are negated
- The TDI, TMS, and TCK TAP inputs are ignored (when in power-on reset or JCOMP not set for NPC operation only
- Registers default back to their reset values

64.1.3.2 Disabled-Port Mode

In disabled-port mode, output pin port enable signals are negated, thereby disabling message transmission. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through $\overline{\text{EVTO}}$.

64.1.3.3 Reduced-Port Mode

Reduced-port mode (RPM) is entered by asserting the MCKO_EN bit and negating the FPM bit in the PCR. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

64.1.3.4 Aurora Link Mode

In Aurora Link mode, trace data is transmitted through the Nexus Aurora Link (NAL) instead of the auxiliary port. For SoCs that support a Nexus Aurora Link, the FPM bit in the PCR functions as the Aurora Link enable, leaving RPM as the only available mode for data transmission via the auxiliary port. Aurora Link Mode is enabled by setting the MCKO_EN and FPM bits and selecting a non-zero value for the ALC_NUM_LN bits in the PCR. The MCKO_DIV value should be set to select a clock frequency equal to the system clock frequency (MCKO_DIV = 000).

64.2 External signal description

64.2.1 Overview

The NPC pin interface provides for the transmission of messages from Nexus blocks to the external development tools and for access to Nexus client registers. The NPC pin definition is outlined in the following table. Pull information for the JTAG signals can be found in the JTAG chapter.

Table 64-1. NPC Signal Properties

Name	Port	Function	Reset State
EVTO_B	Nexus	Event Out pin	0b1
JCOMP	JTAG	JTAG Compliancy and TAP Sharing Control	—
MDO	Auxiliary	Message Data Out pins	0[1]
MSEO	Auxiliary	Message Start/End Out pins	0b11
TCK	JTAG	Test Clock Input	—
TDI	JTAG	Test Data Input	—
TDO	JTAG	Test Data Output	High Z[2]
TMS	JTAG	Test Mode Select Input	—

[1] Following a power-on reset, MDO[0] remains asserted until power-on reset is exited and the system clock achieves lock.

[2] TDO output buffer enable is negated when the NPC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented on TDO at the SoC level.

64.2.2 Detailed signal descriptions

This section describes each of the signals listed in Table 64-1 in more detail. Note that the JTAG test clock (TCK) input from the pin is not a direct input to the NPC. The NPC requires two separate input clocks for TCK clocked logic, one for posedge (rising edge TCK) logic and one for negedge (falling edge TCK) logic. Both clocks are derived from the pin TCK, and generated external to the NPC.

64.2.2.1 EVTO_B — Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus blocks that implement the signal.

64.2.2.2 JCOMP — JTAG Compliancy

The JCOMP signal provides the ability to share the TAP. The NPC TAP controller remains in reset until it is enabled using the JCOMP signal.

64.2.2.3 MDO — Message Data Out

Message Data Out (MDO) are output pins used for uploading OTM, BTM, DTM, and other messages to the development tool. The development tool should sample MDO on the rising edge of MCKO. The width of the MDO bus used is determined by reset configuration.

64.2.2.4 MSEO_B — Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is an output pin that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool should sample $\overline{\text{MSEO}}$ on the rising edge of MCKO.

64.2.2.5 TCK — Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

64.2.2.6 TDI — Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

64.2.2.7 TDO — Nexus Test Data Output

Test Data Output (TDO) pin transmits serial output for instructions and data. TDO is three-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

64.2.2.8 TMS — Test Mode Select

Test Mode Select Input (TMS) pin is used to sequence the IEEE 1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

64.3 Register definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

The following table shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC block does not implement the client select control register because the value does not matter when accessing the registers.

Note that the bypass and instruction registers have no index values. These registers are not accessed in the same manner as Nexus client registers. Refer to the individual register descriptions for more detail.

Table 64-2. NPC registers

Index	Register
0	Device ID Register (DID)
126	Aurora Configuration Register (ACR)
127	Port Configuration Register (PCR)

64.3.1 Register descriptions

This section consists of NPC register descriptions.

64.3.1.1 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

64.3.1.2 Instruction register

The NPC block uses a 4-bit instruction register as shown in the following table. The instruction register is accessed via the SELECT_IR_SCAN path of the tap controller state machine, and allows instructions to be loaded into the block to enable the NPC for register access (NEXUS_ENABLE) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in synchronous loading of the BYPASS instruction. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	3	2	1	0
R	Previous Instruction Opcode			
W	Instruction Opcode			
Reset:	BYPASS Instruction Opcode (0xF)			

64.3.1.3 Nexus Device ID (DID) Register

The device identification register, shown in the following figure, allows the part revision number, design center, part identification number, and manufacturer identity code of the part to be determined through the output port.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Part Revision Number				Design Center						Part Identification Number					
W																
Reset	PRN				DC						PIN					

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Part Identification Number				Manufacturer Identity Code											1
W																
Reset	PIN (cont'd)				0	0	0	0	0	0	0	1	1	1	0	1

Table 64-5. DID field descriptions

Bit	Name	Description
31:28	PRN	Part Revision Number These bits contain the revision number of the part
27:22	DC	Design Center These bits indicate the device design center
21:12	PIN	Part Identification Number These bits contain the part number of the device
11:1	MIC	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID for Freescale Semiconductor, 0xE.
0	Bit [0]	IDCODE Register ID Bit [0] identifies this register as the device identification register and not the bypass register

64.3.1.4 Port Configuration Register (PCR)

Register index: 127

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FPM	MCKO_GT	MCKO_EN	MCKO_DIV			EVT_EN	Reserved	Reserved	0	0	0	ALC_NUM_LN		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LP_DBG	0	0	0	0	0	LP2_SYN	LP1_SYN	0	0	0	0	0	0	0	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCR, shown in the figure above, is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

The PCR register may be rewritten by the debug tool subsequent to the enabling of the NPC for low power debug support. In this case, the debug tool may set and clear the LP_DBG and LPn_SYN bits, but must preserve the original state of the remaining bits in the register. While enabling the low power mode handshake (setting LP_DBG), the debugger should simultaneously set the LPn_SYN bit. At the end of low-power mode handshake, the LP_DBG and LPn_SYN should be set again if further handshakes between the device and debugger are required.

NOTE

The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

Table 64-7. PCR field descriptions

Bit	Name	Description
31	FPM	Full Port Mode The value of the FPM bit determines if Nexus Aurora mode is enabled or if the MDO port is used to transmit messages. 0 A subset of MDO pins are used to transmit messages 1 Nexus Aurora mode enabled
30	MCKO_GT	MCKO Clock Gating Control This bit is used to enable or disable MCKO clock gating. If clock gating is enabled, the MCKO clock is gated when the NPC is in enabled mode but not actively transmitting messages on the auxiliary output port. When clock gating is disabled, MCKO is allowed to run even if no auxiliary output port messages are being transmitted. 0 MCKO gating is disabled 1 MCKO gating is enabled
29	MCKO_EN	MCKO Enable This bit enables the MCKO clock to run. When enabled, the frequency of MCKO is determined by the MCKO_DIV field. 0 MCKO clock is driven to zero 1 MCKO clock is enabled
28:26	MCKO_DIV	MCKO Division Factor The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. Table 64-8 shows the meaning of MCKO_DIV Values. In this table, SYS_CLK represents the system clock frequency. Set to 000 when Nexus Aurora mode is enabled.
25	EVT_EN	EVTO/EVTI Enable This bit enables the EVTO/EVTI port functions. 0 EVTO/EVTI port disabled 1 EVTO/EVTI port enabled
24	Reserved	
23	Reserved	
19:18	ALC_NUM_LN	Aurora Lane Control, Number of Lanes The value of this bit determines how many high-speed Aurora lanes are enabled 00 0 Lanes Enabled 01 2 Lanes Enabled 10 Reserved 11 Reserved
15	LP_DBG_EN	Low Power Debug Enable

Table continues on the next page...

Table 64-7. PCR field descriptions (continued)

Bit	Name	Description
		This bit enables debug functionality on exit from low power modes on supported devices. 0 Low power debug disabled 1 Low power debug enabled
9	LP2_SYN	Low Power Mode 2 Synchronization These bits are used to synchronize the entry into low power modes between the device and debug tool. Supported devices set these bits before a pending entry into low power mode. After reading the bit as set, the debug tool then clears the bit to acknowledge to the device that it may enter the low power mode. 0 Low power mode entry acknowledged 1 Low power mode entry pending
8	LP1_SYN	Low Power Mode 1 Synchronization These bits are used to synchronize the entry into low power modes between the device and debug tool. Supported devices set these bits before a pending entry into low power mode. After reading the bit as set, the debug tool then clears the bit to acknowledge to the device that it may enter the low power mode. 0 Low power mode entry acknowledged 1 Low power mode entry pending
0	Reserved	

Table 64-8. MCKO_DIV Values

MCKO_DIV[2:0]	MCKO Frequency
0	SYS_CLK[1]
1	SYS_CLK/2
2	Reserved
3	SYS_CLK/4
4	Reserved
5	Reserved
6	Reserved
7	Reserved

[1] SYS_CLK setting for MCKO frequency should only be used if this setting does not violate the maximum operating frequency of the auxiliary port pins.

64.3.1.5 Aurora Configuration Register (ACR)

Register index: 126

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LVDS_RXOPT[3:0]				0	0	0	0	LVDS_TXOPT[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The ACR, shown in the figure above, is used to configure the RXOPT and TXOPT settings of the LVDS receive and transmit pads.

Table 64-10. ACR Field Descriptions

Bit	Name	Description
15:12	LVDS_RXOPT[3:0]	Configures LVDS receive pad options rxopt3, rxopt2, rxopt1, rxopt0
7:0	LVDS_TXOPT[7:0]	Configures LVDS transmit pad options txopt7, txopt6, txopt5, txopt4, txopt3, txopt2, txopt1, txopt0

64.4 Functional Description

64.4.1 NPC reset configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the output port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

For SoCs that support message transmission via the Nexus Aurora Link, asserting the MCKO_EN and FPM bits and setting the NUM_ALC_LN bits to a non-zero value enables NAL data transmission. Transmission of data via the auxiliary port is enabled only by selecting the RPM configuration in this case.

The following table describes the NPC reset configuration options.

Table 64-11. NPC reset configuration options

Power On Reset status asserted?	JCOMP asserted high	MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register	NUM_ALC_LN bits of the PCR > 0	Configuration
yes	X	X	X	X	Reset
no	no	X	X	X	Reset
no	yes	0	X	X	Disabled
no	yes	1	0	X	Reduced Port Mode
no	yes	1	1	yes	Aurora Link Mode

64.4.2 Auxiliary Output Port

The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the Nexus modules and arbitrates for access to the port.

64.4.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the $\overline{\text{MSEO}}$ functions. The $\overline{\text{MSEO}}$ pins are used to signal the end of variable-length packets and the end of messages. They are not required to indicate the end of fixed-length packets. MDO and $\overline{\text{MSEO}}$ are sampled on the rising edge of MCKO.

The following figure illustrates the state diagram for $\overline{\text{MSEO}}$ transfers. All transitions not included in the figure are reserved, and must not be used.

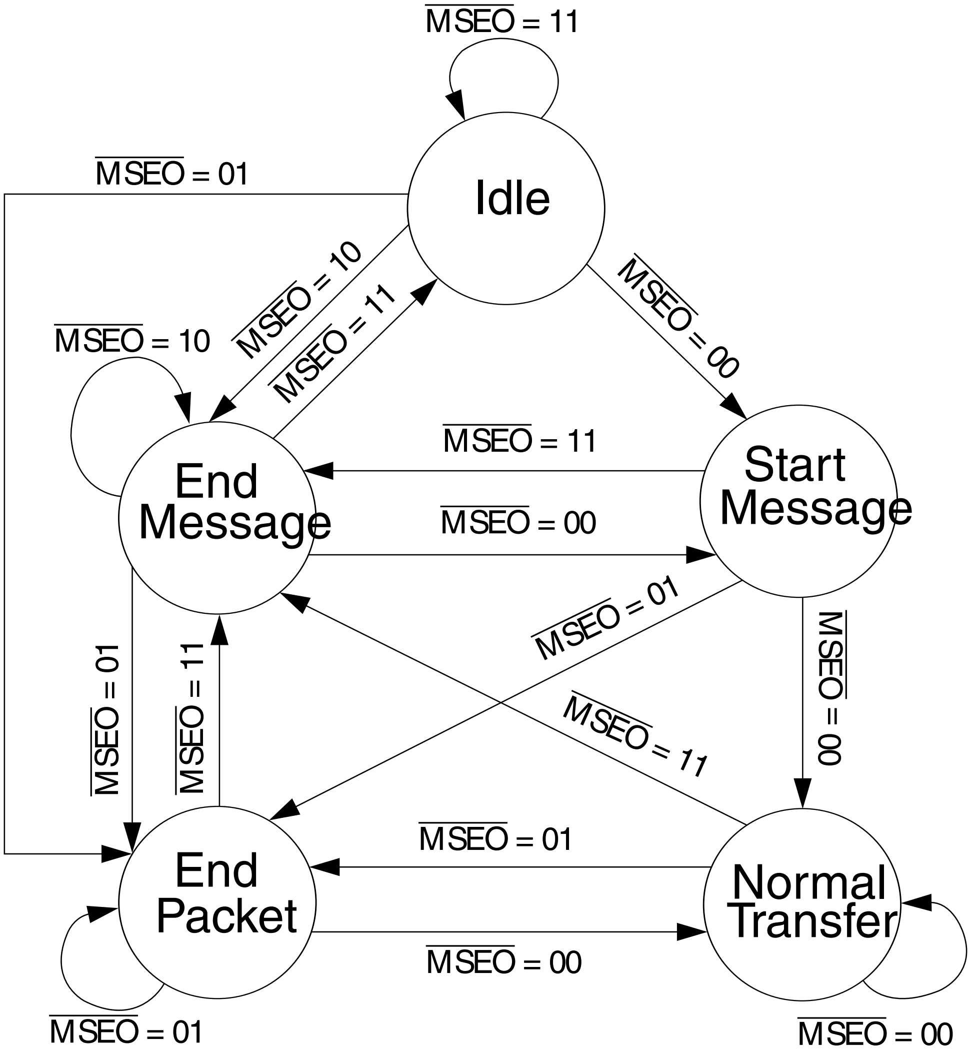


Figure 64-2. $\overline{\text{MSEO}}$ Transfers (for 2-bit $\overline{\text{MSEO}}$)

64.4.2.2 Output Messages

In addition to sending out messages generated in other Nexus blocks, the NPC can also output the device ID message contained in the device ID register on the MDO pins. The device ID message can also be sent out serially through TDO.

Figure 64-3 describes the device ID output message that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Device ID Message	6	6	fixed	TCODE	Value = 1
	32	32	fixed	ID	DID register contents
Aurora Link Overrun Error Message[1]	6	6	fixed	TCODE	Value = 8
	4	4	fixed	SRC	Nexus Aurora Link SRC value. Set during SoC integration via plug.
	5	5	fixed	ECODE	Value = 01000. Error code indicating overrun of NAL FIFO.

NOTES:

[1] The Aurora Link Overrun Error Message only transmits when there is a NAL overrun in Nexus Aurora Link Mode. This message is not valid in modes where data is transmitted via the auxiliary port.

Figure 64-3. NPC output messages

64.4.2.3 Rules of Message

- A variable-sized field within a message must end on a port boundary. (Port boundaries depend on the number of MDO pins active with the current reset configuration.)
- A variable-sized field may start within a port boundary only when following a fixed-length field.
- Super-fields must end on a port boundary.
- When a variable-length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary.
- Multiple fixed-length packets may start and/or end on a single clock.
- When any packet follows a variable-length packet, it must start on a port boundary.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information.
- Within a field, the lowest significant bits are shifted out first. The following figure shows the transmission sequence of a message that is made up of a TCODE followed by two fields.

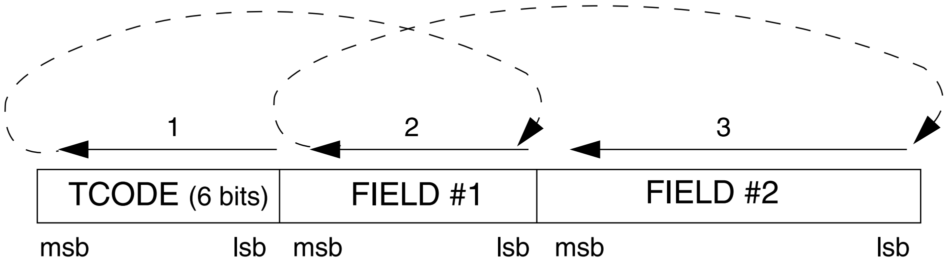


Figure 64-4. Transmission Sequence of Messages

64.4.3 IEEE 1149.1-2001 (JTAG) TAP

The NPC block uses the IEEE 1149.1-2001 test access port (TAP) for accessing registers. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. There may also be other blocks on the MCU that use the TAP and implement a TAP controller. .

Refer to the IEEE 1149.1-2001 specification for further detail on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1-2001 state machine shown in Figure 64-6 . The Nexus controller state machine is defined by the IEEE-ISTO 5001-2001 standard. It is shown in Figure 64-7.

The instructions implemented by the NPC TAP controller are listed in the following table. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4 bits.

Table 64-12. Implemented Instructions

Instruction Name	Private/Public	Opcode	Description
NEXUS-ENABLE	public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

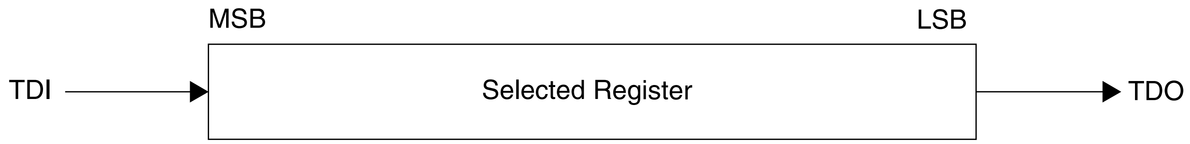
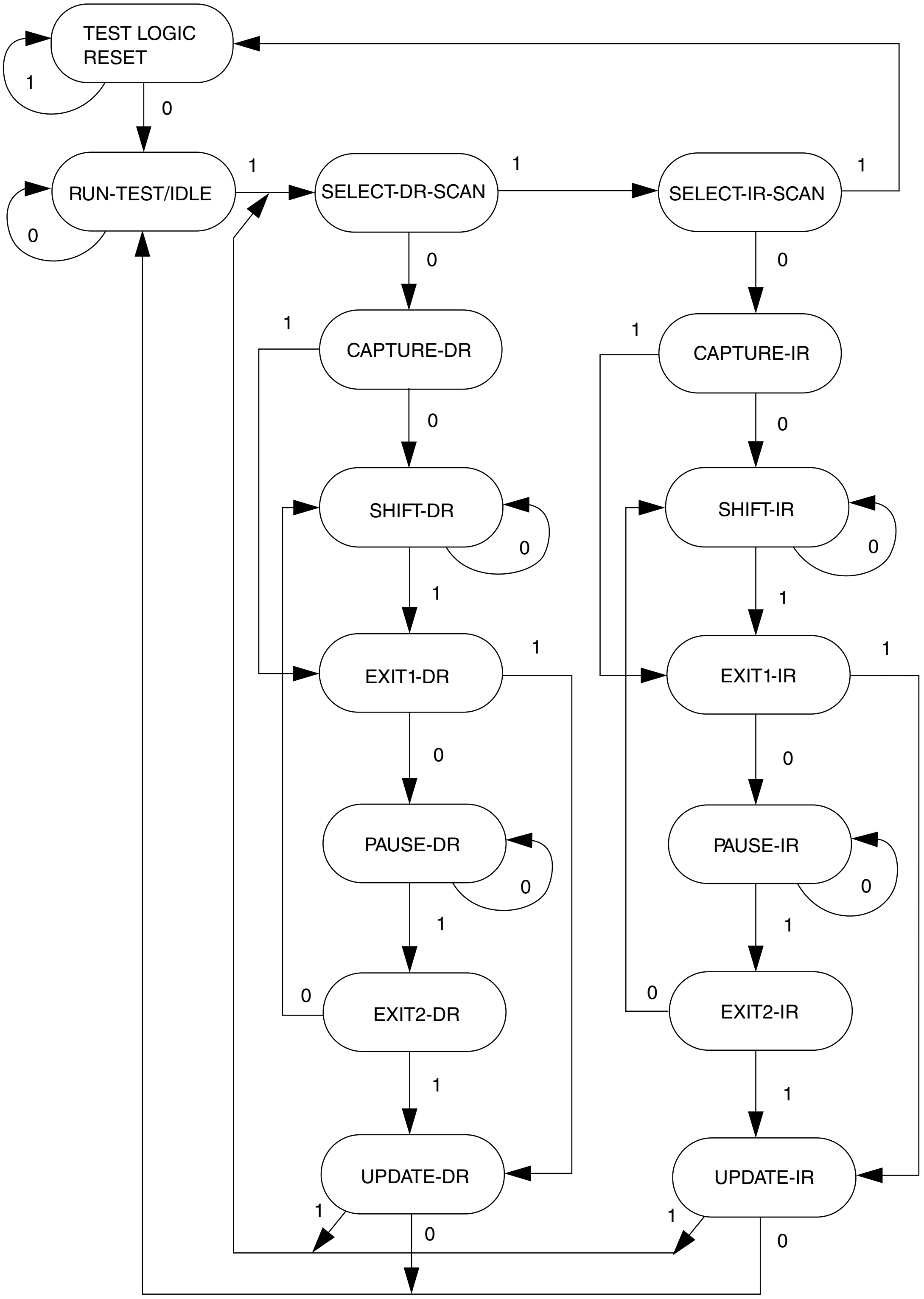


Figure 64-5. Shifting Data Into Register

64.4.3.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal or setting JCOMP to a value other than the NPC enable encoding resets the NPC TAP controller. When not in power-on reset, the NPC TAP controller is enabled by driving JCOMP with the NPC enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 64-6. IEEE 1149.1-2001 TAP Controller State Machine

64.4.3.2 Retrieving device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the output port or shifted out serially by accessing the Nexus Device ID register through the TAP.

Transmission of the device identification message on the auxiliary output port MDO pins occurs immediately after a write to the PCR, if the NPC is enabled for auxiliary port mode. Transmission of the device identification message serially via the TAP is achieved by performing a read of the register contents as described in Selecting a Nexus Client Register. The device identification message is not transmitted on the Aurora Link in Aurora Link mode, and it can be retrieved only via the TAP in that mode.

64.4.3.3 Loading NEXUS-ENABLE Instruction

Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 64-7, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. Table 64-13 illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

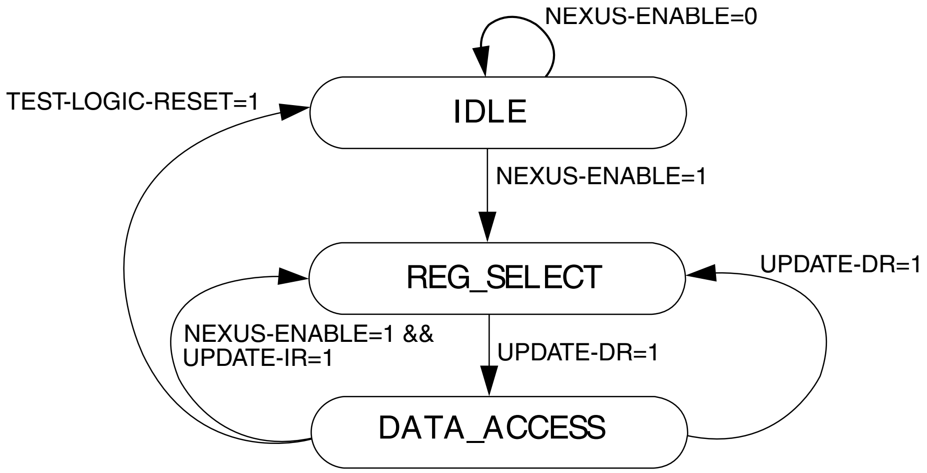


Figure 64-7. NEXUS Controller State Machine

Table 64-13. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state

Table continues on the next page...

Table 64-13. Loading NEXUS-ENABLE instruction (continued)

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
3 TCKS				
8	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
9	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
10	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

64.4.3.4 Selecting a Nexus Client Register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following table. The read/write control bit is set to 1 for writes and 0 for reads.

MSB	LSB
7-bit register index	R/W

Figure 64-8. IEEE 1149.1 Controller Command Input

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the JTAG shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 64-14. Write to a 32-Bit Nexus Client Register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

64.4.4 Nexus JTAG Port Sharing

Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register.

64.4.5 MCKO Control

MCKO is an output clock to the development tools used for the timing of $\overline{\text{MSEO}}$ and MDO pin functions. MCKO is derived from the system clock and its frequency is determined by the value of the MCKO_DIV field in the PCR. Possible operating frequencies include system clock, one-half system clock, one-quarter system clock, and one-eighth system clock speed.

The NPC also generates an MCKO clock gating control output signal. This output can be used by the MCKO generation logic to gate the transmission of MCKO when the auxiliary port is enabled but not transmitting messages. The setting of the MCKO_GT bit inside the PCR determines whether or not MCKO gating control is active. The MCKO_GT bit resets to a logic 0. In this state gating of MCKO is disabled. To enable gating of MCKO, the MCKO_GT bit in the PCR is written to a logic 1.

64.4.6 $\overline{\text{EVTO}}$ Sharing

The NPC block controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. The NPC assumes incoming $\overline{\text{EVTO}}$ signals will be asserted for one system clock period. After receiving a single clock period of asserted $\overline{\text{EVTO}}$ from any Nexus client, the NPC latches the result, and drives $\overline{\text{EVTO}}$ for one MCKO period on the following clock. When there is no active MCKO, such as in disabled mode, the NPC drives $\overline{\text{EVTO}}$ for two system clock periods. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

64.4.7 Nexus Reset Control

The JCOMP input that is used as the primary reset signal for the NPC is also used by the NPC to generate a single-bit reset signal for other Nexus blocks. If JCOMP is negated, an internal reset is asserted, indicating that all Nexus modules should be held in reset.

64.4.8 System Clock Locked Indication

Following a power-on reset, MDO[0] can be monitored to provide the lock status of the system clock. MDO[0] is driven to a logic 1 until the system clock achieves lock after exiting power-on reset. Once the system clock is locked, MDO[0] is negated and tools may begin Nexus configuration. Loss of lock conditions that occur subsequent to the exit of power-on reset and the initial lock of the system clock do not cause a Nexus reset, and therefore do not result in MDO[0] driven high.

64.4.9 Data Transmission via Nexus Aurora Link

The NPC supports data transmission over a high-speed serial Nexus Aurora Link (NAL). The NPC's role in Aurora Link data transmission is to route trace data to the NAL while continuing to manage arbitration between Nexus clients for the right to transmit data. In addition, the NPC monitors the NAL transmit queue and will halt further data transmission from Nexus clients if the queue is in danger of an overrun condition. In the event of an overrun condition, the NPC transmits a standard Nexus error message.

Once the PCR is configured for Aurora Link Mode (MCKO_EN bit set, FPM bit set, ALC_NUM_LN set to non-zero value), the NPC waits for the NAL to acknowledge it is ready to receive data before starting data transmission. All data transmitted in Aurora Link mode is transmitted in frames, with each frame consisting of 8192 bytes of data. The first beat of each frame is accompanied by the assertion of the start of frame control signal, and the last beat of each frame is accompanied by the assertion of the end of frame control signal. Each beat of valid NAL data from the NPC is also accompanied by the assertion of the NAL data valid message signal.

Data received by the NAL is held in a FIFO until it can be sent out on the Aurora Link. The NPC keeps track of the NAL FIFO fill level. To help prevent an overrun of the NAL FIFO, the NPC holds off further grants to all clients if the NAL FIFO is filled above the watermark level. In the event that the NAL FIFO is overrun with data, the NPC transmits its FIFO overrun error message shown in Figure 64-3.

64.5 Initialization/Application Information

64.5.1 Accessing NPC tool-mapped registers

To initialize the TAP for Nexus register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Load the TAP controller with the NEXUS-ENABLE instruction

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.
2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.

2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE-ISTO 5001-2001 standard for more detail.

Chapter 65
Nexus Aurora Link (NAL)

65.1 Introduction

The Aurora Protocol is used to send debug information over a high-speed serial link out of the device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer SerDes and transceivers; it includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol. The following figure details the NAL block diagram. The number of lanes, n, is 2.

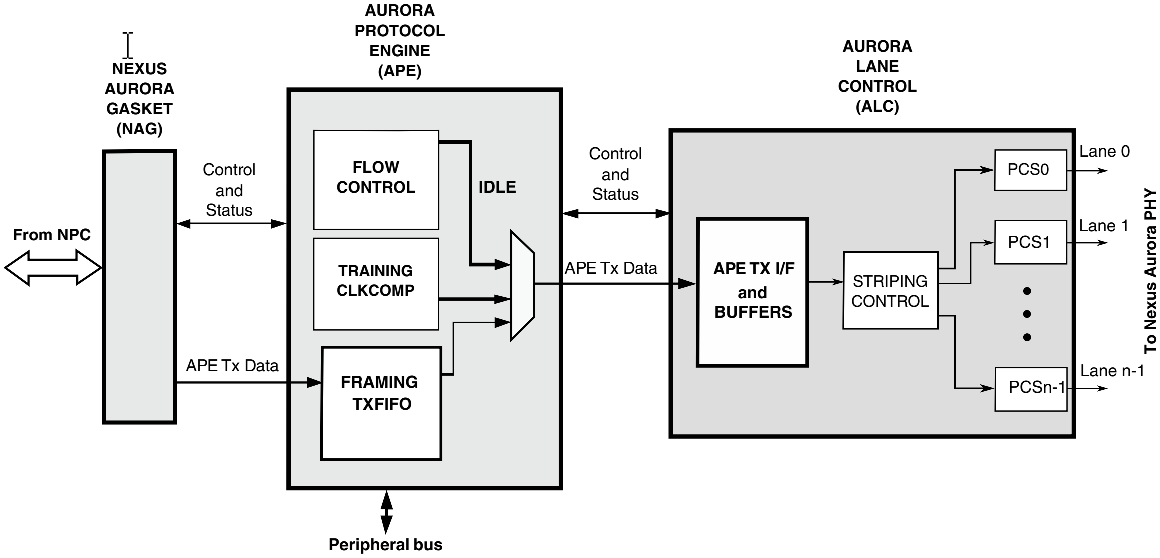


Figure 65-1. NAL block diagram

Debug data in the Nexus format from multiple clients is collected by the Nexus Port Controller (NPC) ; messages are then packetized and sent across an asynchronous clock boundary-crossing gasket to the Aurora Protocol Engine, which performs all the logical layer functions and flow control; data from the protocol engine is sent on to the Aurora Lane Control (ALC) block, which performs data-striping and sizing, based on the number of lanes selected for debug; it also performs the PCS functions of the link. The PCS drives the Nexus Aurora Physical (NAP) interface. The NAP, in conjunction with the NAL, supports one-way simplex high-speed serial communication with an external debugger.

The NAL is composed of three main blocks: the NPC/Aurora Gasket, the Aurora Protocol Engine, and Aurora Lane Control. The table below lists NAL functionality by block and lists the high level features for the Nexus Aurora PHY.

Table 65-1. NAL and NAP feature summary

Sub-Component	Feature
NPC Aurora Gasket (NAG)	• Transmit FIFO to buffer data as it crosses the NPC/NAL clock-domain boundary and Transmit FIFO space available indication to NPC. The NPC uses Transmit FIFO space available indication(s) to determine if data can be sent to the NAL. • Platform to Domain clock domain crossing. The NPC is on the core clock domain while the NAP is on a domain sourced from the external PHY clock input signals.
Aurora Protocol Engine (APE)	• Link training • Per the Aurora Protocol specification, clock-compensation, and data framing
Aurora Link Control (ALC)	• Striping of data stream from the Aurora core across all available lanes in two-byte chunks • PCS functionality for NAL
Nexus Aurora PHY (NAP)	• Physical transmission of serial data to device pins • External clock source for SerDes clock domain

65.2 Transmit operation

During transmit operation, the NPC collects debug data from Nexus clients over a 32-bit interface: 30 bits are used for MDO and 2 bits are used for MSEO information. The NPC places MDO data into a payload (maximum payload size depends on the number of lanes, each lane has a maximum of 7424 symbols before terminating the payload) and adds the following:

- 16-bit start symbol
- 16-bit end symbol

Each payload is transferred from the NPC to the NPC/NAL gasket, through the Aurora Protocol Engine, the Aurora Lane Control and finally out to the device SerDes/LVDS pins through the NAP.

After each end symbol packet transfer, the NAL adds a clock compensation sequence needed for downstream SerDes devices to remain synchronized to the MCU.

NOTE

Symbol counting for clock compensation starts just prior to the verification symbols; therefore, training symbols are excluded when counting.

65.3 Nexus Aurora formatting

The Aurora physical interface is the transport mechanism for Nexus messages. The Nexus messages are formatted by the Aurora formatter and striped into Aurora lanes. The Nexus messages themselves encapsulated into a virtual Nexus port that consists of a 30-bit Message Data Output (MDO) port and a 2-bit Message Start/End Output (MSEO) signal. The MSEO portion of the port is the two least significant bits (LSB) of the 32-bit Aurora message. The rest of the 32-bit Aurora frame consists of 30 MDO bits that are added LSB first.

65.4 Static training

The NAL supports static training. The NAL performs Static training by using programmable timers to determine how long to stay in each stage of the training process. In static training, there is no communication between the channel partners to exchange training status; rather the NAL sends the align/bond/verify sequence to its partner for a set amount of time sufficient for the receive partner to complete the training stage.

Note

Prior to initialization, the NAL must be reset to its POR state; to do this, assert the block reset. See the RST bit field description in the NAL General Control Register (NAL_GCR).

The following steps enable static training:

1. Write 80000000h to NAL_GCR
2. Write 00000000h to NAL_GCR
3. Write 80783C0Fh to NAL_TCR
4. Write 80000000h to NAL_GCR
5. Write 00000000h to NAL_GCR
6. Write A0408000h to NPC_PCR

This resets the channel and start the training routine. The time spent in each stage can range from 16 to 3072 cycles; once a counter times out, a counter time-out signal is asserted to the Aurora Protocol Engine (APE) to prompt it to move to the next stage.

Additionally, it is possible to hold the training procedure in any of the stages indefinitely, by setting one of the three hold bits (AHD/BHD/VHD). When set, these bits cause the channel to sit in Align/Bond/Verify until unset, allowing a user to interactively step through the training procedure by monitoring the status at the debugger side and then stepping the device side to the next stage.

During the Aurora training sequence, the NAL generates a specific set of character sequences per the Aurora protocol specification which are to be transmitted simultaneously on all lanes.

65.5 Clock compensation considerations

The Aurora Protocol Specification describes clock compensation as a means for the link to compensate for clock rate differences between the transmitter and receiver. The spec requires that a clock compensation be inserted at a minimum every 10000 symbol pairs; this should allow the link to accomodate up to a 200 ppm difference in clock rates.

Due to the fact that the NAL is restricted to only inserting clock-compensation sequences after an end-of-frame, it may not be possible to always meet the Aurora spec requirement. However, there are ways to control the framing of data through the link and insure that satisfactory performance is maintained. There are two types of scenarios that can prevent an adequate number of clock compensation sequences from being generated: the sparse data case and the unframed data case. In the sparse data case, there are long periods of time when no data is flowing through the link; no data means no frames. In the unframed case, there is plenty of data, but this data is not framed, and so again, clock-compensation never gets a chance to make it out.

The unframed case is simplest to address. The NPC can either operate in framed or unframed mode. In unframed mode, an end-of-frame is only injected when the NPC runs out of data (as soon as new data is collected, it is sent out accompanied with a start-of-frame; if the NPC is configured in this mode, the clock-compensation spec can easily be violated. However, in framed mode, an end-of-frame/start-of-frame pair is injected after every 256 data transmissions. Here, each frame consists of 256 x AW symbol pairs. Inside the APE, a clock-compensation counter keeps track of how much data has been transmitted to the ALC since the last clock-compensation sequence; once the counter hits 7424, clock-compensation will be sent on the next end-of-frame.

On the other hand, even if the NPC is in framing mode, there is little that can be done if the NPC has no data to send. In the sparse data scenario, the NAL could end up transmitting idles for an undetermined number of cycles while waiting for the NPC to provide new data. However, a user can get around this problem by insuring that the NPC always has at least some minimal amount of data to send. There are several ways to do this:

- If full-duplex infrastructure is available, then the simplest thing to do is for the debugger to monitor the TX data stream from the NAL for long idle periods. If the link has been idle for longer than some minimal amount of time (say 8000 symbol pairs), then the debugger could send in a read request through the aurora-in RX port -- a quick NRR read of the dev_id register would be ideal for this purpose. The NPC would then return the requested data, accompanied by a start-of-frame/end-of-frame pair, which the NAL could then tag with clock-compensation. See the NPC block guide for more details on aurora-in NRR access.
- If full-duplex is not available, then the debugger could still force the NPC to send the dev_id message by going through the JTAG/SAP port and setting the NPC SRESET bit. Reinitializing the NPC puts it back into its startup state, wherein it will invariably send device ID before doing anything else. The only drawbacks to this approach are its long latency (going through JTAG) and the fact that NPC must be correctly reconfigured after every SRESET.
- Another approach is to use periodic watchpoints to insure that the data stream never falls completely silent for more than some predefined count. Most nexus clients, including the NPC itself, can generate a watchpoint sync message based on an external or internal trigger. By hooking one of these triggers up to a counter (a performance-monitor, event-processing unit debug counter can be used for this purpose), we can insure that, even if there is no other trace data traffic, there will at least be periodic watchpoint messages which can be used to trigger clock compensation. The drawback here is that it requires devoting one of the debug counters for clock compensation, but this may not always be a problem: one of the proposed methods of correlating nexus timestamps from different asynchronous domains is to use the same sort of periodic sync trigger to force all asynchornous clients to send a stamped watchpoint message at (roughly) the same time. This asynchronous sync trigger can also be used for clock compensation.

65.6 Programmable registers

The Nexus Aurora Link (NAL) module provides a Nexus parallel to Aurora formatting for the MCU trace information. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 65-2. NAL memory map

Offset	Register name	Width (in bits)	Access	Reset value
0h	NAL General Status Register (NAL_GSR)	32	R	See section
8h	NAL General Control Register (NAL_GCR)	32	R/W	0000_0000h
Ch	NAL Training Control Register (NAL_TCR)	32	R/W	0000_0000h

65.6.1 Nexus Aurora Link General Status Register (NAL_GSR)

The NAL_GSR is a read only register that describes the status of the NAL. See the chip-specific NAL information for the NAL_GSR reset value.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												Reserved	0		
W																

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			TXCFG			0									AS
W																

Table 65-3. NAL_GSR field descriptions

Field	Description
31–20 Reserved	Reserved This bitfield is reserved.
19 Reserved	Reserved This bitfield is reserved.
18–13 Reserved	Reserved This bitfield is reserved.
12–10 TXCFG	TX Lane Configuration. Number of enabled TX lanes. These bits are read-only and are controlled external to the NAL. 000 Reserved 001 2 TX lanes 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
9–1 Reserved	Reserved This bitfield is reserved.

Table continues on the next page...

Table 65-3. NAL_GSR field descriptions (continued)

Field	Description
0 AS	Aurora Status. 0 Aurora is not enabled. 1 Aurora is enabled.

65.6.2 NAL General Control Register (NAL_GCR)

The NAL_GCR configures the behavior of the APE. It can be programmed to reset the Aurora channel, or to configure simplex/duplex initialization and to enable debug features.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RST	ALCPD	0												FCM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PCRST	0	SLD			0	BOOE	BOOD				0	CCOEN	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 65-4. NAL_GCR field descriptions

Field	Description
31 RST	Reset the Aurora Channel. 0 to 1 transition resets the channel and begins the training sequence. 0 normal operation 1 transition from 0 to 1 resets channel and starts training sequence
30 ALCPD	ALC/NAL Power Down. Gate off clocks to the NAL blocks (ALC and APE). 0 NAL is enabled with all clocks running provided other mechanisms of powerdown (such as alc_num_lanes) are not in effect. 1 NAL is powered down and inactive.
29–18 Reserved	Reserved This read-only bitfield is reserved and always has the value zero.
17 FCM	Flow Control Mode. 0 Completion Mode 1 Immediate Mode
16-15	Reserved and must be written with zero.

Table continues on the next page...

Table 65-4. NAL_GCR field descriptions (continued)

Field	Description
Reserved	
14 PCRST	Protocol Converter Reset. Resets the ALC when asserted.
13 Reserved	Reserved This read-only bit is reserved and always has the value zero.
12–10 SLD	Symbol Lock Delay. 000 Allow 3 characters to accumulate in elastic buffer before starting to read 001 Allow 4 characters to accumulate in elastic buffer before starting to read 010 Allow 5 characters to accumulate in elastic buffer before starting to read 011 Allow 6 characters to accumulate in elastic buffer before starting to read 100 Allow 7 characters to accumulate in elastic buffer before starting to read 101 Allow 8 characters to accumulate in elastic buffer before starting to read 110 Allow 9 characters to accumulate in elastic buffer before starting to read 111 Allow 10 characters to accumulate in elastic buffer before starting to read
9 Reserved	Reserved This read-only bit is reserved and always has the value zero.
8 BOOE	Bond Offset Override Enable. 0 ALC channel bond attempts to determine channel bond offset. 1 ALC uses user override setting for channel bond offset.
7–4 BOOD	Bond offset override data. Only valid when BOOE is set. 0000/1000 use 0-cycle offset 0001 use -7 cycle offset 0010 use -6 cycle offset 0011 use -5 cycle offset 0100 use -4 cycle offset 0101 use -3 cycle offset 0110 use -2 cycle offset 0111 use -1 cycle offset 1001 use +1 cycle offset 1010 use +2 cycle offset 1011 use +3 cycle offset 1100 use +4 cycle offset 1101 use +5 cycle offset 1110 use +6 cycle offset 1111 use +7 cycle offset
3 Reserved	Reserved and must be written with zero.

Table continues on the next page...

Table 65-4. NAL_GCR field descriptions (continued)

Field	Description
2 CCOEN	Clock Compensation Override Enable. This is a debug feature which allows the duration between clock compensation sequences to be reduced to 1280 cycles. 0 Clock compensation counter times out at 7424 cycles 1 Clock compensation counter times out at 1280 cycles
1–0 Reserved	Reserved This read-only bitfield is reserved and always has the value zero.

65.6.3 NAL Training Control Register (NAL_TCR)

The NAL_TCR supports static training as described in Static training.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STE	AHD	BHD	VHD	Reserved					ATC				Reserved					BTC				Reserved				VTC					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 65-5. NAL_TCR field descriptions

Field	Description
31 STE	Static Training Enable. 0 Reserved 1 Timer-based training method is used, ie. RX data is ignored and training is established based on ATC/BTC/VTC timers.
30 AHD	Hold in Align. 0 Follow Aurora training sequence 1 Remain in align until this bit is cleared (applies even during duplex-mode training)
29 BHD	Hold in Bond. 0 Follow Aurora training sequence 1 Remain in bond until this bit is cleared. (applies even during duplex-mode training)
28 VHD	Hold in Verify. 0 Follow Aurora training sequence 1 Remain in verify until this bit is cleared. (applies even during duplex-mode training)
27–23 Reserved	Reserved This bitfield is reserved and must be written with zero.
22–19 ATC	Align Timer Count 0000 Remain in align for 16 cycles 0001 Remain in align for 24 cycles

Table continues on the next page...

Table 65-5. NAL_TCR field descriptions (continued)

Field	Description
	0010 Remain in align for 32 cycles 0011 Remain in align for 48 cycles 0100 Remain in align for 64 cycles 0101 Remain in align for 96 cycles 0110 Remain in align for 128 cycles 0111 Remain in align for 192 cycles 1000 Remain in align for 256 cycles 1001 Remain in align for 384 cycles 1010 Remain in align for 512 cycles 1011 Remain in align for 768 cycles 1100 Remain in align for 1024 cycles 1101 Remain in align for 1536 cycles 1110 Remain in align for 2048 cycles 1111 Remain in align for 3072 cycles
18–14 Reserved	Reserved This bitfield is reserved.
13–10 BTC	Bond Timer Count 0000 Remain in bond for 16 cycles 0001 Remain in bond for 24 cycles 0010 Remain in bond for 32 cycles 0011 Remain in bond for 48 cycles 0100 Remain in bond for 64 cycles 0101 Remain in bond for 96 cycles 0110 Remain in bond for 128 cycles 0111 Remain in bond for 192 cycles 1000 Remain in bond for 256 cycles 1001 Remain in bond for 384 cycles 1010 Remain in bond for 512 cycles 1011 Remain in bond for 768 cycles 1100 Remain in bond for 1024 cycles 1101 Remain in bond for 1536 cycles 1110 Remain in bond for 2048 cycles 1111 Remain in bond for 3072 cycles
9–4 Reserved	Reserved This bitfield is reserved.
3–0 VTC	Verify Timer Count 0000 Remain in verify for 16 cycles 0001 Remain in verify for 24 cycles

Table 65-5. NAL_TCR field descriptions

Field	Description
	0010 Remain in verify for 32 cycles 0011 Remain in verify for 48 cycles 0100 Remain in verify for 64 cycles 0101 Remain in verify for 96 cycles 0110 Remain in verify for 128 cycles 0111 Remain in verify for 192 cycles 1000 Remain in verify for 256 cycles 1001 Remain in verify for 384 cycles 1010 Remain in verify for 512 cycles 1011 Remain in verify for 768 cycles 1100 Remain in verify for 1024 cycles 1101 Remain in verify for 1536 cycles 1110 Remain in verify for 2048 cycles 1111 Remain in verify for 3072 cycles

Chapter 66
Nexus Aurora PHY (NAP)

66.1 Introduction

The Nexus Aurora PHY (NAP), in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. Configuration settings in the Nexus Port Controller (NPC) Port Configuration Register (NPC_PCR) specify the NAP and NAL lane configuration. The NAL performs all logical layer functions, flow control, data-striping and sizing based on the number of data lanes available for use, and physical coding sublayer functions on the data being sent to the NAP for transmission. The NAP receives 8b/10b encoded data from the NAL then serializes and transmits this data through a given PHY lane's Low Voltage Differential Signaling (LVDS) buffers to an external debugger.

Each NAP lane consists of a data symbol character accumulator which registers the 10-bit character for that lane from the NAL on the rising edge of the NAL clock. Next, a serializer takes the 10-bit character and serializes it into the individual data bits that are then transmitted at the PHY clock rate via the LVDS output drivers to the external debugger receive channel.

Note

> The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called Lane Alignment in Aurora terminology), 10b/8b decoding, decode and disparity error tracking in the received data stream.

The following figure shows a high level block diagram of the High Speed Nexus Interface and the NAP.

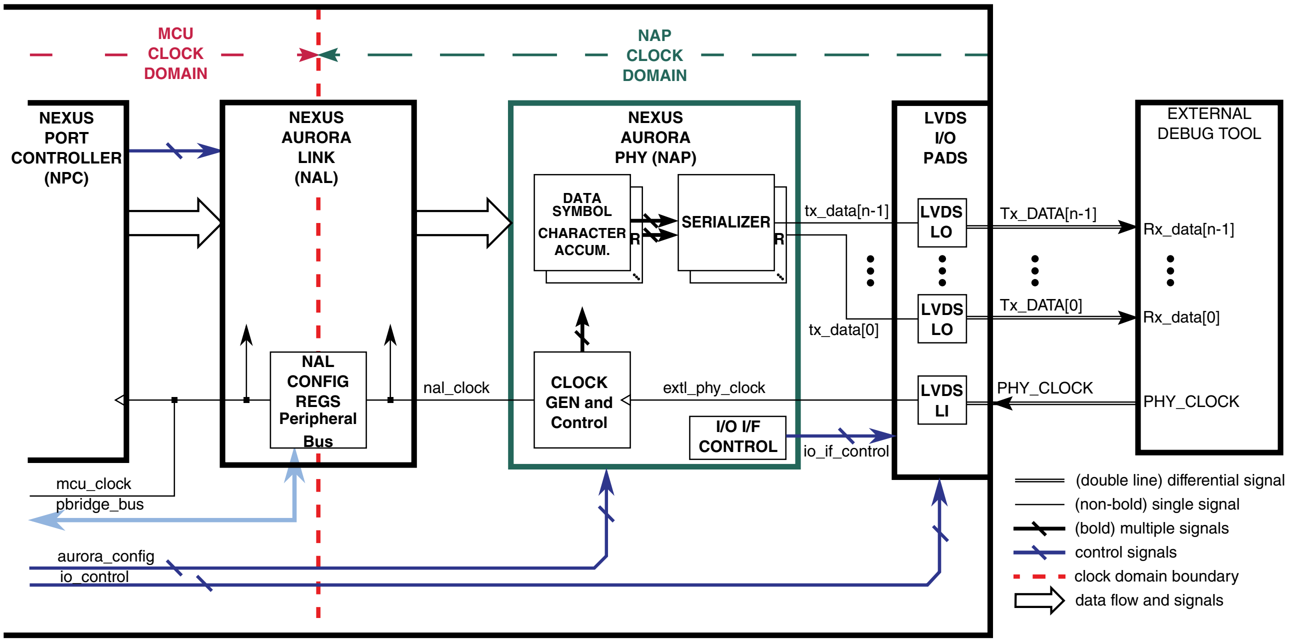


Figure 66-1. High speed Nexus interface block diagram

66.1.1 Features

The features of the NAP are as follows:

- Xilinx Aurora Protocol Specification V2.x compliant
- 1Gb/s per channel trace data payload bandwidth with 1.25 GHz clock
- Simplex mode with static (timer-based) channel training

66.1.2 Modes of operation

The operating mode of the NAP is determined by the device control signals. The functional configuration is determined by the NPC_PCR setting; functional operation is enabled by the NPC.

66.1.2.1 Reset

During Nexus Reset all internal NAP resources are reset, all output ports are disabled, and all inputs are ignored. Since there is no device clock domain logic in the NAP itself, the NAP is unaffected by device system reset. Depending on the NAP configuration settings, the NAP may activate upon exiting Nexus Reset if any Aurora lanes are enabled during this reset mode, else it remains disabled and inactive.

66.1.2.2 Link training

Before beginning trace output mode operation, the Aurora Link must go through a training process conducted by the NAL to ensure that an external debugger can correctly reconstruct the transmitted data stream. Training proceeds once the link powers up, and must complete successfully before data transmission can occur.

The NAL initializes and trains the Aurora Link in a simplex configuration using a timer based static-training method. The initialization and training sequence described in Aurora Protocol Specification progresses from one step to the next via the use of programmable timers contained in the NAL. A timer is programmed for each stage of training and as each timer expires, the training sequence proceeds to the next stage until complete. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

66.1.2.3 Transmit

During Nexus trace output operation, it is the responsibility of the NPC to push data into the Aurora Link at a sufficiently high rate to ensure that the link does not become data-starved. The NAL and NAP support up to 2 transmit lanes. The Aurora encapsulated data is modified by the NAL to ensure the appropriate number of symbols are transmitted through the NAP.

66.2 External signal description

The external signals of the NAP are listed in the following table. Each of the NAP signals consists of a positive and negative differential pair.

Table 66-1. NAP signals

Signal Name	Direction	Description
PHY_CLOCK (CLKP/CLKN)	Input	NAP clock input from external debug tool

Table continues on the next page...

Table 66-1. NAP signals (continued)

Signal Name	Direction	Description
TX Data0 (TX0P/TX0N)	Output	NAP Tx data for lane 0
TX Data1 (TX1P/TX1N)	Output	NAP Tx data for lane 1

66.3 Memory map and register definition

There are no control, configuration or status registers within the NAP. All control, configuration and status options for number of lanes and LVDS pads are handled in the NPC_PCR which is described in the NPC Chapter.

66.4 Functional description

High-level descriptions of the operation of the NAP and its main functional blocks are given in the following sections.

66.4.1 Data symbol character accumulator

The data symbol character accumulator portion of the NAP for each lane is made up of a symbol character register whose inputs are connected to the 10-bit NAL character bus. The data symbol character accumulator essentially registers a 10-bit character for that lane simultaneous with the rising edge of the NAL clock and presents the 10-bit quantity to the serializer at the appropriate times.

66.4.2 Serializer

The serializer portion of the NAP for each lane takes the data captured by that lane's Data Symbol Character Accumulator and demuxes the data to provide a single serial stream of symbol data to be output by the LVDS LO buffer for that lane at the PHY clock rate.

The following figure shows a high level timing diagram that demonstrates the operation of the serializer for a NAP lane, where S0–S9 represents a 10-bit symbol, S10–S19 represent the next symbol, and so on.

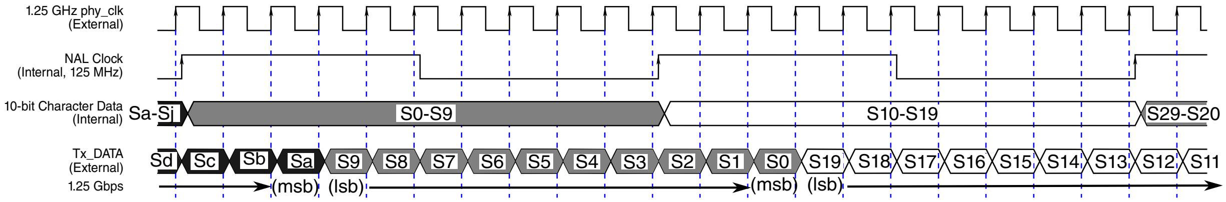


Figure 66-2. Serializer timing diagram

66.4.3 Clock generation and control

The Clock Generation and Control block is responsible for generating the necessary clocks and state signals needed by the rest of the NAP. The Data Symbol Character Accumulators and serializers are the main consumers of the signals generated by the Clock Generation and Control block.

66.5 Initialization information

The NAP does not require initialization other than the Nexus Reset and programming of the NPC_PCR to configure the number of lanes and LVDS pads.

Chapter 67
Cyclic Redundancy Check (CRC) Unit

67.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of three hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/ halfword/word) formats.

67.2 Main features

- 3 contexts for the concurrent CRC computation are supported
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- Three hard-wired polynomials (CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)
- Support for byte, halfword, or word width of the input data stream

67.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN

- CRC-16-CCITT
- CRC-32 Ethernet (see note below)

NOTE

Although the CRC-32 Ethernet polynomial is used for CRC generation, the generation algorithm differs (MISR) from the one used by the Ethernet protocol (IEEE 802).

67.3 Block diagram

The top level diagram of the CRC module is given in the following figure. Refer to Main features for number of contexts in this module.

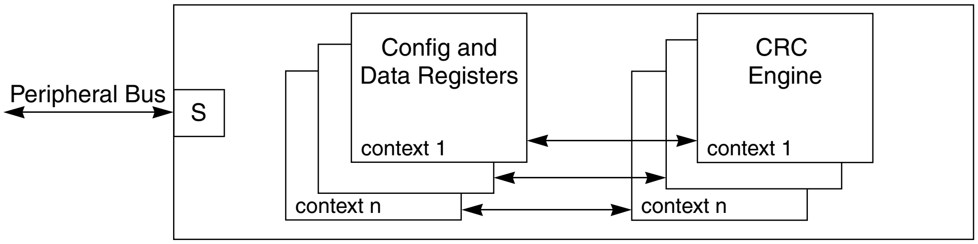


Figure 67-1. CRC top level diagram

67.4 External signal description

The CRC module does not generate any external signals.

67.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers

- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers
- Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are read/write accessible in each access mode:

- user
- supervisor

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_ CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_ CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

67.5 CRC memory map and registers

CRC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Configuration Register (CRC_CFG1)	32	R/W	See section	67.5.1/2678
4	Input Register (CRC_INP1)	32	R/W	0000_0000h	67.5.2/2679
8	Current Status Register (CRC_CSTAT1)	32	R/W	FFFF_FFFFh	67.5.3/2680
C	Output Register (CRC_OUTP1)	32	R	FFFF_FFFFh	67.5.4/2680
10	Configuration Register (CRC_CFG2)	32	R/W	See section	67.5.1/2678
14	Input Register (CRC_INP2)	32	R/W	0000_0000h	67.5.2/2679
18	Current Status Register (CRC_CSTAT2)	32	R/W	FFFF_FFFFh	67.5.3/2680

Table continues on the next page...

CRC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C	Output Register (CRC_OUTP2)	32	R	FFFF_FFFFh	67.5.4/2680
20	Configuration Register (CRC_CFG3)	32	R/W	See section	67.5.1/2678
24	Input Register (CRC_INP3)	32	R/W	0000_0000h	67.5.2/2679
28	Current Status Register (CRC_CSTAT3)	32	R/W	FFFF_FFFFh	67.5.3/2680
2C	Output Register (CRC_OUTP3)	32	R	FFFF_FFFFh	67.5.4/2680

67.5.1 Configuration Register (CRC_CFG*n*)

Access: User read/write.

Address: 0h base + 0h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SWAP_ BYTEWISE	SWAP_BITWISE	POLYG		SWAP	INV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	*	*	0	0

* Notes:

- POLYG field: Reset value is a function of the number of contexts, N. When N%2 = 0, reset value = 01b. When N%2 = 1, reset value = 00b.

CRC_CFG*n* field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SWAP_ BYTEWISE	Swap CRC_INP byte-wise

Table continues on the next page...

CRC_CFG*n* field descriptions (continued)

Field	Description
	NOTE: INV and SWAP bits are set to 1 for CRC-32 polynomial calculations. 0 Do not swap 1 Perform byte-wise swap on CRC_INP input data internally for CRC-32 polynomial calculations.
27 SWAP_BITWISE	Swap CRC_INP bit-wise 0 Do not swap 1 Perform bit-wise swap on CRC_INP input data internally for CRC-8 and CRC-16 polynomial calculations.
28–29 POLYG	Polynomial selection This bit can be written only during the configuration phase. 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 Reserved
30 SWAP	Swap selection This bit can be written only during the configuration phase. The swap operation is a bit-by-bit swapping of the content. 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content; in case of CRC-CCITT polynomial, the swap operation is applied on the 16 least-significant bits
31 INV	Inversion selection This bit can be written only during the configuration phase. The inversion operation is a complement (or negation) of the content. 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content

67.5.2 Input Register (CRC_INP*n*)

Access: User read/write.

Address: 0h base + 4h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRC_INP*n* field descriptions

Field	Description
0–31 INP	Input data for the CRC computation

CRC_INP*n* field descriptions (continued)

Field	Description
	This register can be written with word-, halfword- (high or low), or byte-wide writes in any sequence. Only the bits written are fed to the CRC engine. In case of halfword write operation, the bytes must be contiguous.

67.5.3 Current Status Register (CRC_CSTAT*n*)

Access: User read/write.

Address: 0h base + 8h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSTAT																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_CSTAT*n* field descriptions

Field	Description
0–31 CSTAT	CRC signature status This register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 most-significant bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be written only during the configuration phase.

67.5.4 Output Register (CRC_OUTP*n*)

Access: User read-only.

Address: 0h base + Ch offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTP																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_OUTP*n* field descriptions

Field	Description
0–31 OUTP	Final CRC signature

CRC_OUTP*n* field descriptions (continued)

Field	Description
	This register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 MSB bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation.

67.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less or equal to the number of contexts.

The standard generator polynomials are given in Equation 56 on page 2681, Equation 57 on page 2681, and Equation 58 on page 2681 for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 56. CRC8 VDA CAN

$$x^{16} + x^{12} + x^5 + 1$$

Equation 57. CRC-CCITT (x.25 protocol)

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Equation 58. CRC-32 (Ethernet protocol)

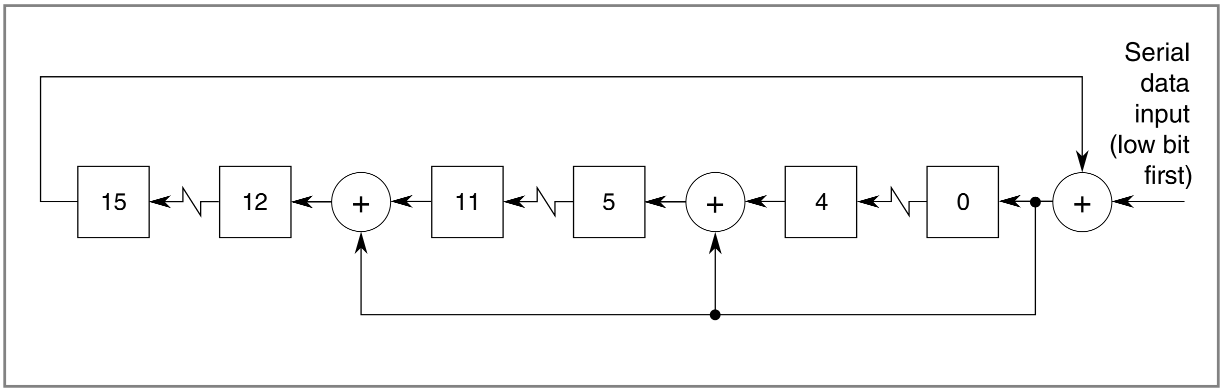


Figure 67-2. CRC-CCITT engine concept scheme

The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. A diagram illustrating serial data loading of the CRC engine is shown in Figure 67-2 for the CRC-CCITT. Note that, conceptually, the least-significant, or "right-most," bit shown in Input Register (CRC_INP*n*), is fed first into the engine. The actual implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the IPS bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (for example, SPI, etc.), a bit swap (high bit for low bit or low bit for high bit) and/or bit inversion of the final CRC signature can be applied to CRC_OUTP before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in Figure 67-3.

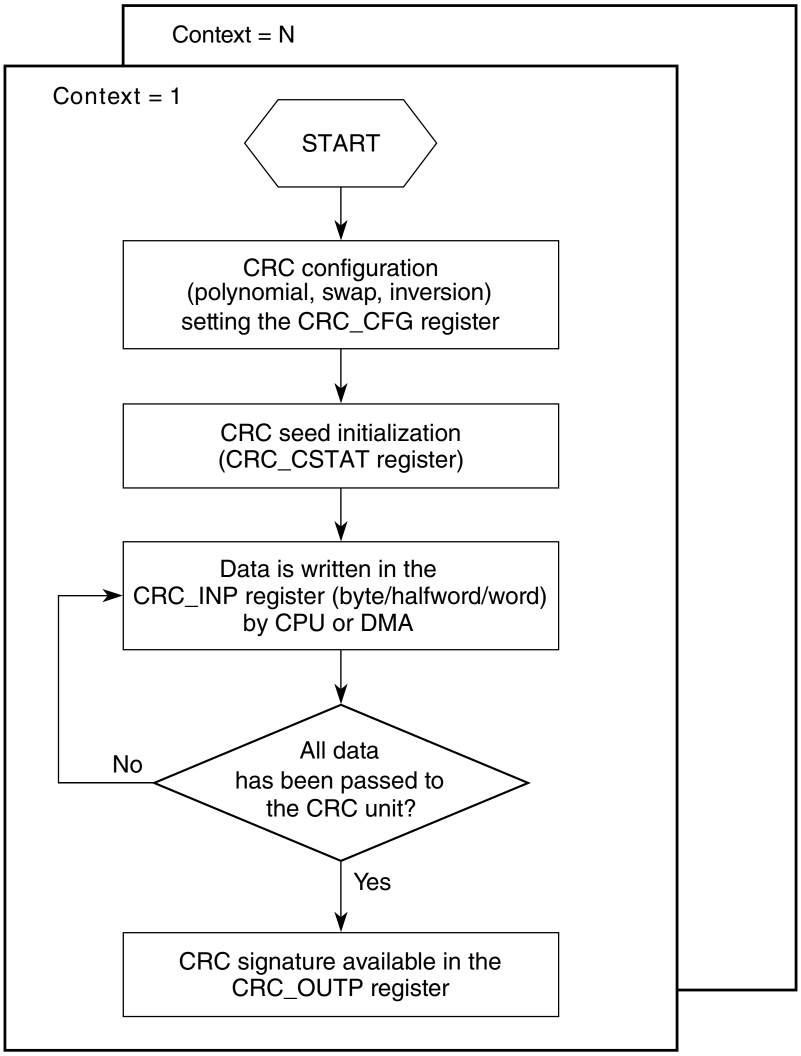


Figure 67-3. CRC computation flow

67.7 Use cases

67.7.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has *n* channels (greater than or equal to the number of contexts) configurable for the following types of data transfer—mem2mem, periph2mem, mem2periph—the following sequence, as shown in the following figure, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x, channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB-to-LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

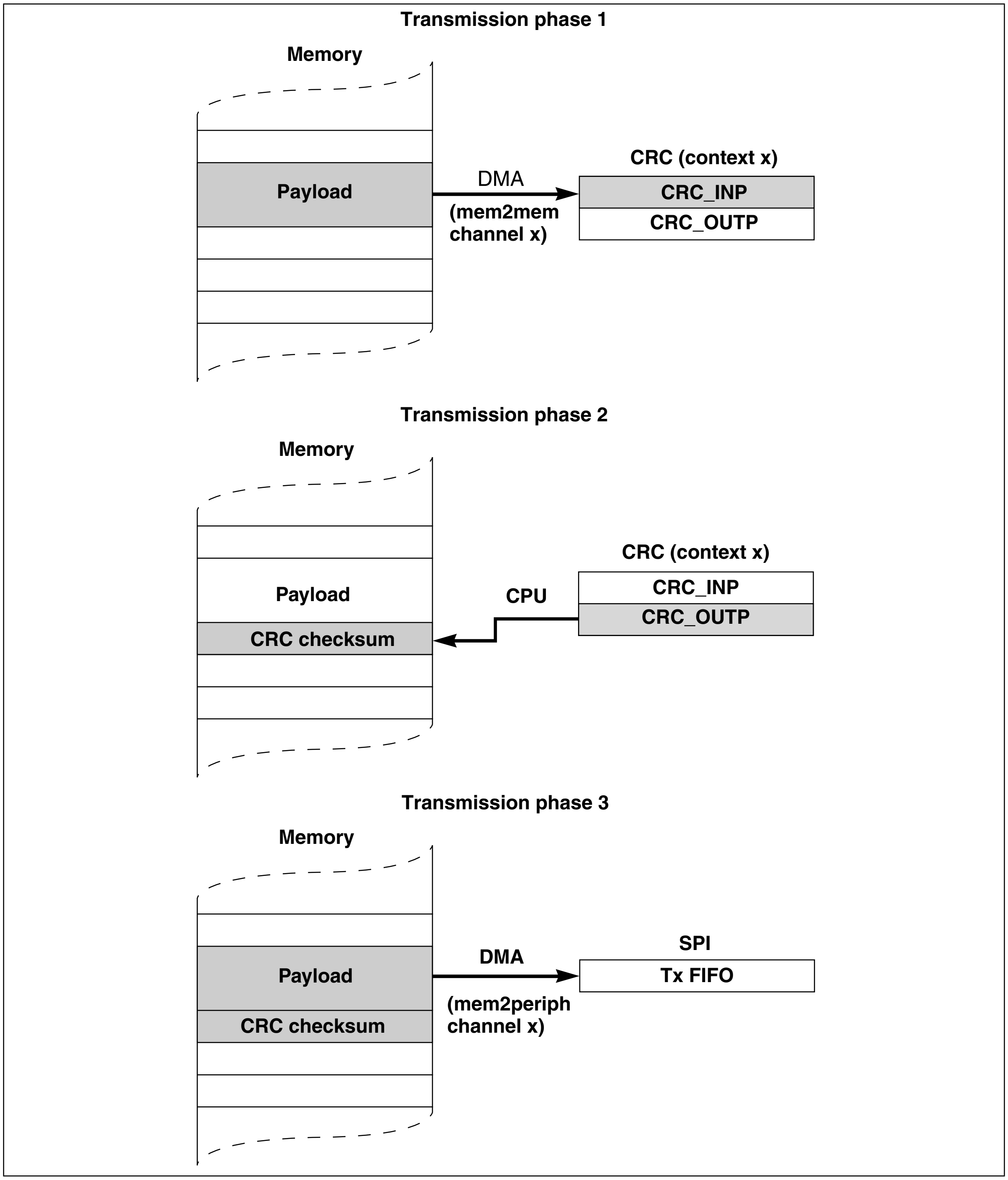


Figure 67-4. DMA-CRC transmission sequence

67.7.2 Register programming

- INV and/or SWAP (affecting the output only) can be applied to CRC-8, CRC-16, and CRC-32. Both can be applied together.

- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the Ethernet standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

Chapter 68
Memory Error Management Unit (MEMU)

68.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- System RAM
- Peripheral RAM
- Flash memory

When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
  - Single Bit error in the data part that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Single Bit error in the data part that is detected via MBIST on any RAM.
- Uncorrectable Error: It comprises the following:
  - Multiple bit error that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Multiple bit error that is detected via MBIST on any SRAM. Based on MBIST implementation, the MBIST might report several correctable errors in the same

word instead of an uncorrectable one. It is the task of the software after MBIST to aggregate these reports and detect the uncorrectable error.
    - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see the chip-specific MEMU information.

When multiple errors are indicated from various sources at same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from either ECC or MBIST logic), independent of the source generating these errors.

The MEMU has multiple instances of the basic reporting block; one each for:

- System RAM ECC and MBIST
- Peripheral RAM ECC
- Flash memory ECC

Each instance has two reporting tables. Correctable errors are reported in one table, and the uncorrectable errors are reported in the other.

See Design overview for the reporting block details.

68.2 Features

The MEMU has the following features:

- Supports up to 128 error sources per the following reporting blocks (the number of error sources are chip-specific; see the chip-specific MEMU information):
    - System RAM ECC and MBIST
    - Peripheral RAM ECC
    - Flash memory ECC
- Support for error reporting from the following category of error sources (both for ECC and MBIST):
    - System RAM
    - Peripheral RAM
    - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.

- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.
- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

68.3 Block diagram

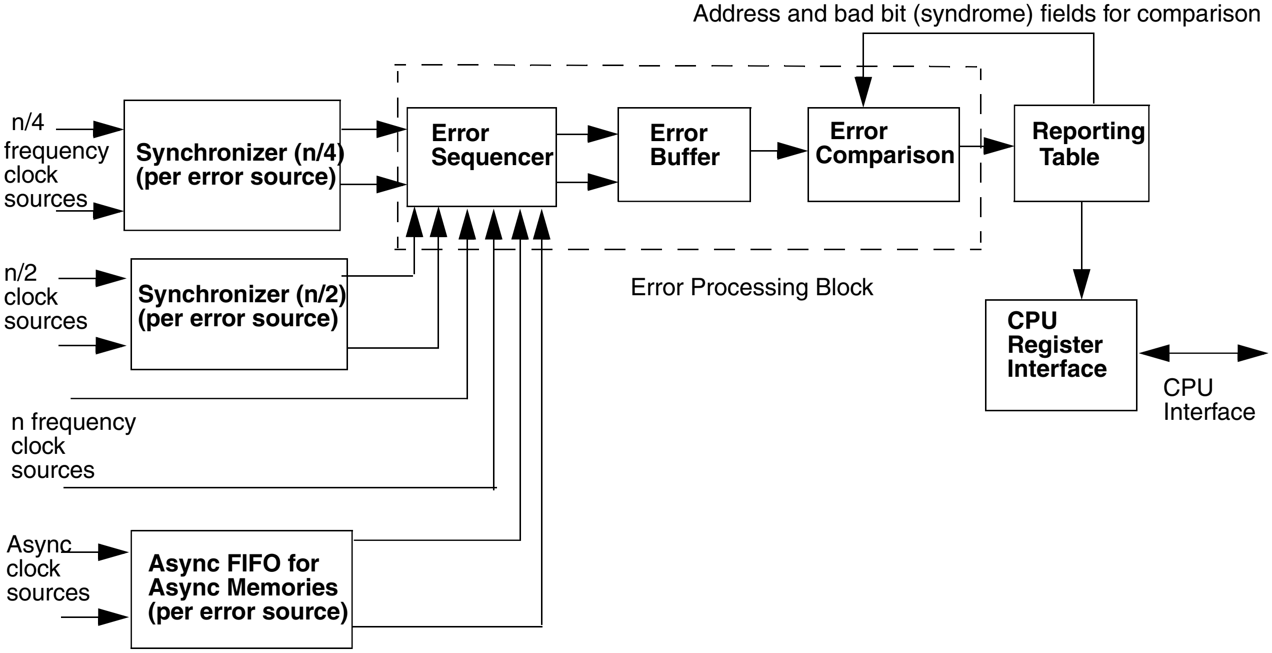

Address and bad bit (syndrome) fields for comparison
n/4 frequency clock sources
Synchronizer (n/4) (per error source)
Error Sequencer
Error Buffer
Error Comparison
Reporting Table
n/2 clock sources
Synchronizer (n/2) (per error source)
Error Processing Block
CPU Register Interface
CPU Interface
n frequency clock sources
Async clock sources
Async FIFO for Async Memories (per error source)

Figure 68-1. MEMU block diagram

68.4 Design overview

The MEMU should be enabled on power-on reset. The user software can write known error addresses into the reporting table to prevent reporting of those errors addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See Handling overflows (Multiple error reporting), for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

   For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

   For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

2. Error Processing block

   This block is used to process the incoming errors coming from the device.

   - Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.
   - All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.
   - While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit/syndrome (depending on ECC or MBIST logic) of the incoming error with the one stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set., else the incoming error is dropped.

- While doing the error buffer uniqueness check, the type of error i.e Correctable Error (CE) or Uncorrectable Error UCE) is also compared, hence even if the Error Address and the Bad bit/syndrome fields (Depending on whether the incoming error is generated from ECC or MBIST logic) of the incoming error matches with the error buffer, if the type of errors (CE and UCE) differ they are treated as unique
- Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable)
- Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (<=5 unless the error gets temporarily stored in the error buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

   This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

   The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

   The CPU can clear the flags signaling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (i.e. ..._CERR_STS or ..._UNCERR_STS). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

   The reporting table sizes are chip-specific; see the chip-specific MEMU information.

4. Register block

   The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signaled to the CPU and the CPU command will be completed in the next clock cycle.

   The software, in principle, is not supposed to write VLD to 1 if it is already 1.

   The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

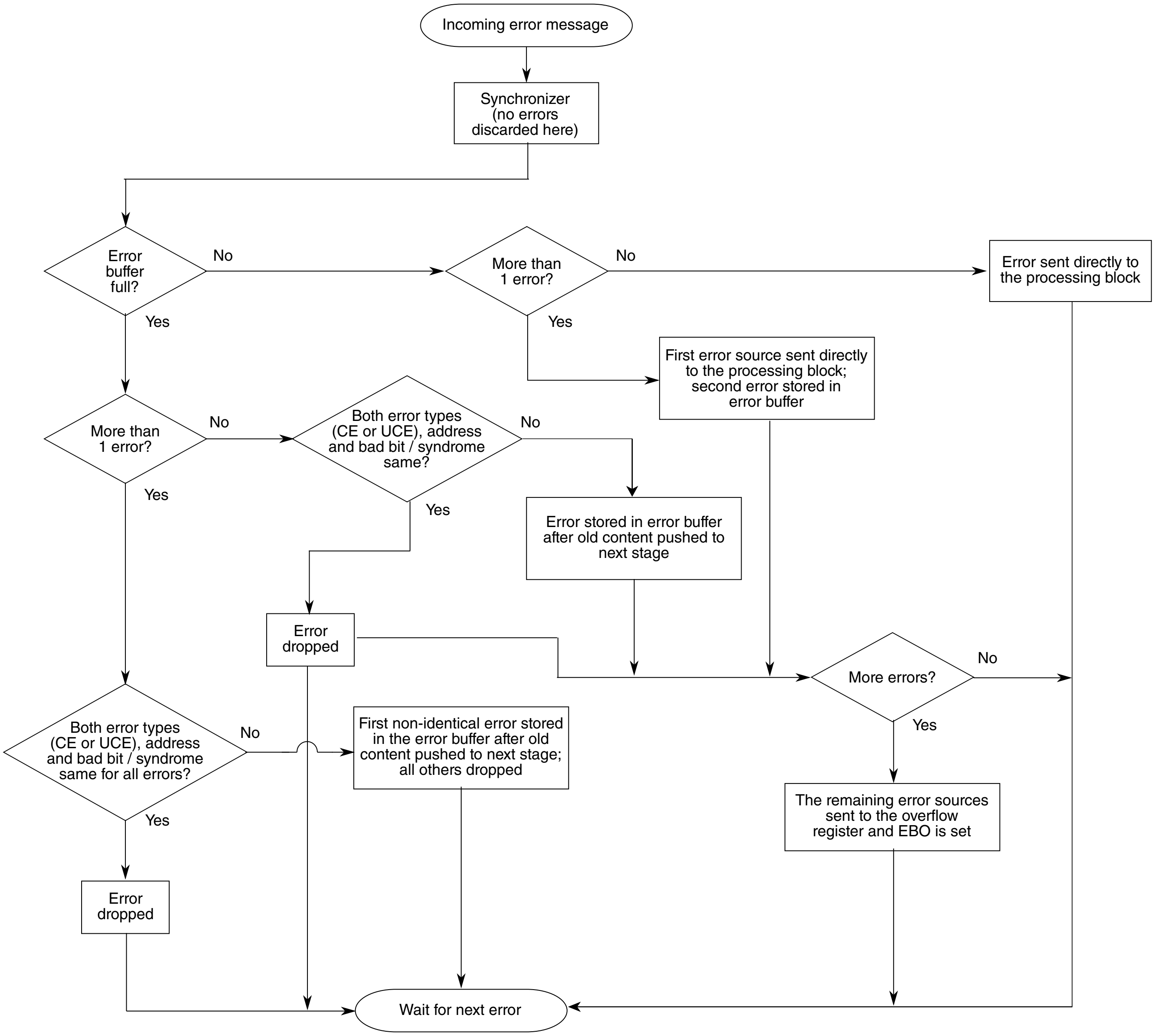

Incoming error message
Synchronizer (no errors discarded here)
Error buffer full?
No
Yes
More than 1 error?
No
Yes
Error sent directly to the processing block
First error source sent directly to the processing block; second error stored in error buffer
More than 1 error?
No
Yes
Both error types (CE or UCE), address and bad bit / syndrome same?
No
Yes
Error stored in error buffer after old content pushed to next stage
Error dropped
More errors?
No
Yes
Both error types (CE or UCE), address and bad bit / syndrome same for all errors?
No
Yes
First non-identical error stored in the error buffer after old content pushed to next stage; all others dropped
The remaining error sources sent to the overflow register and EBO is set
Error dropped
Wait for next error

Figure 68-2. MEMU error buffer uniqueness check flow chart

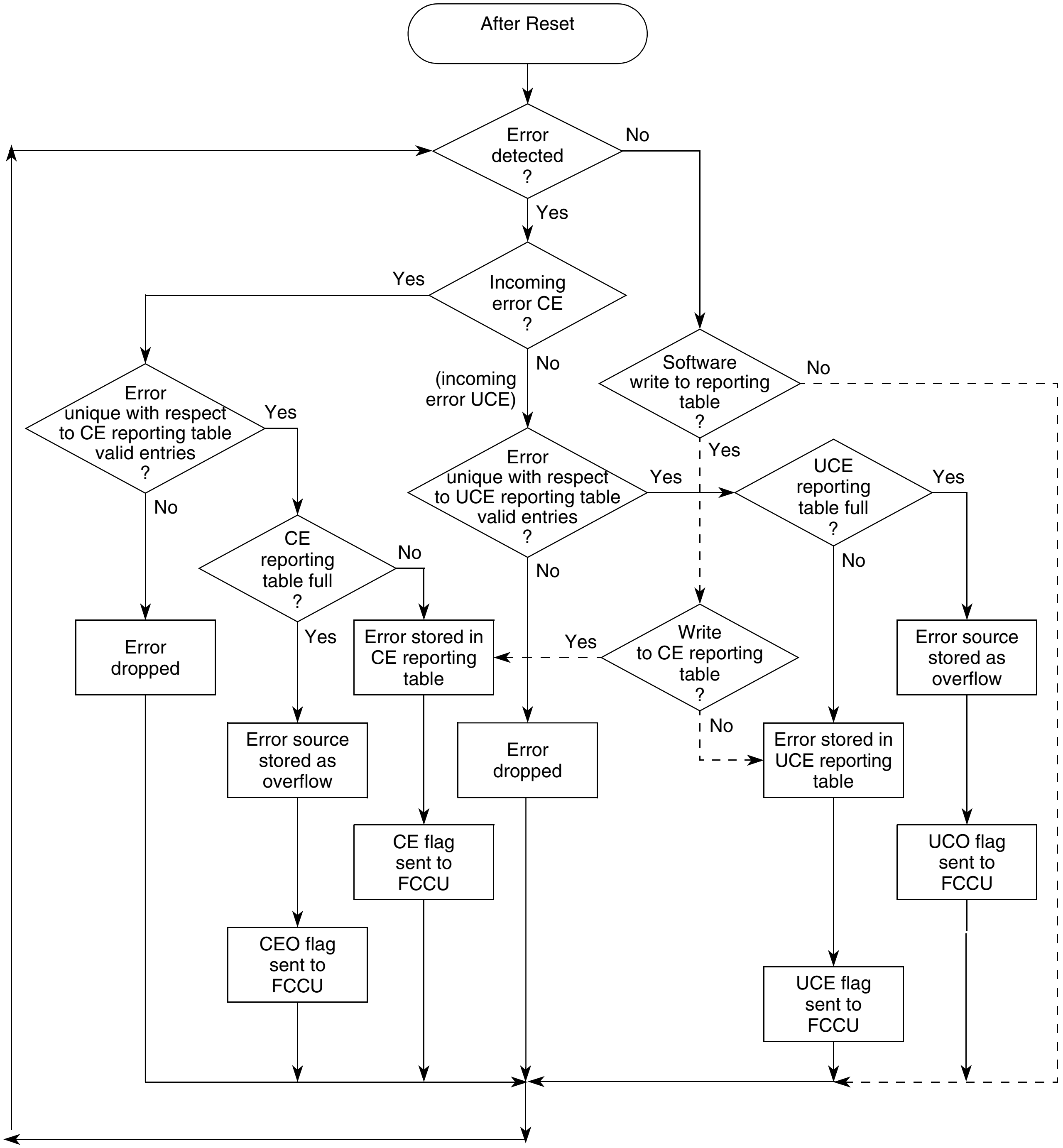

After Reset
Error detected ?
No
Yes
Incoming error CE ?
Yes
No
(incoming error UCE)
Software write to reporting table ?
No
Yes
Error unique with respect to CE reporting table valid entries ?
Yes
No
CE reporting table full ?
No
Yes
Error dropped
Error stored in CE reporting table
Error source stored as overflow
CE flag sent to FCCU
CEO flag sent to FCCU
Error unique with respect to UCE reporting table valid entries ?
Yes
No
Error dropped
UCE reporting table full ?
Yes
No
Write to CE reporting table ?
Yes
No
Error stored in UCE reporting table
UCE flag sent to FCCU
Error source stored as overflow
UCO flag sent to FCCU

Figure 68-3. MEMU reporting table uniqueness check flow chart

68.5 External signal description

MEMU has no external signals.

68.6 Memory map and register definition

This section describes the registers on this module.

68.6.1 Overview

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

NOTE

The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

NOTE

The memory map in this section shows all the possible registers on this module and the associated offsets (which vary depending on the number of registers actually implemented). The actual availability, number and addresses of registers, and number of reported errors in the reporting table are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

MEMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control register (MEMU_CTRL)	32	R/W	0000_0000h	68.6.2/2698
4	Error flag register (MEMU_ERR_FLAG)	32	w1c	0000_0000h	68.6.3/2698
C	Debug register (MEMU_DEBUG)	32	R/W	0000_0000h	68.6.4/2702
20	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS0)	32	R/W	0000_0000h	68.6.5/2704
24	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR0)	32	R/W	0000_0000h	68.6.6/2705
28	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS1)	32	R/W	0000_0000h	68.6.5/2704
2C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR1)	32	R/W	0000_0000h	68.6.6/2705
30	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS2)	32	R/W	0000_0000h	68.6.5/2704

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR2)	32	R/W	0000_0000h	68.6.6/2705
38	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS3)	32	R/W	0000_0000h	68.6.5/2704
3C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR3)	32	R/W	0000_0000h	68.6.6/2705
40	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS4)	32	R/W	0000_0000h	68.6.5/2704
44	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR4)	32	R/W	0000_0000h	68.6.6/2705
48	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS5)	32	R/W	0000_0000h	68.6.5/2704
4C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR5)	32	R/W	0000_0000h	68.6.6/2705
50	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS6)	32	R/W	0000_0000h	68.6.5/2704
54	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR6)	32	R/W	0000_0000h	68.6.6/2705
58	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS7)	32	R/W	0000_0000h	68.6.5/2704
5C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR7)	32	R/W	0000_0000h	68.6.6/2705
60	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS8)	32	R/W	0000_0000h	68.6.5/2704
64	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR8)	32	R/W	0000_0000h	68.6.6/2705
68	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS9)	32	R/W	0000_0000h	68.6.5/2704
6C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR9)	32	R/W	0000_0000h	68.6.6/2705
70	System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)	32	R/W	0000_0000h	68.6.7/2705
74	System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	68.6.8/2706
78	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW0)	32	w1c	0000_0000h	68.6.9/2706
7C	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW1)	32	w1c	0000_0000h	68.6.9/2706
80	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW2)	32	w1c	0000_0000h	68.6.9/2706
620	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS0)	32	R/W	0000_0000h	68.6.10/ 2707
624	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR0)	32	R/W	0000_0000h	68.6.11/ 2708

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
628	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS1)	32	R/W	0000_0000h	68.6.10/ 2707
62C	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR1)	32	R/W	0000_0000h	68.6.11/ 2708
630	Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)	32	R/W	0000_0000h	68.6.12/ 2708
634	Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	68.6.13/ 2709
638	Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW0)	32	w1c	0000_0000h	68.6.14/ 2709
C20	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS0)	32	R/W	0000_0000h	68.6.15/ 2710
C24	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR0)	32	R/W	0000_0000h	68.6.16/ 2711
C28	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS1)	32	R/W	0000_0000h	68.6.15/ 2710
C2C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR1)	32	R/W	0000_0000h	68.6.16/ 2711
C30	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS2)	32	R/W	0000_0000h	68.6.15/ 2710
C34	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR2)	32	R/W	0000_0000h	68.6.16/ 2711
C38	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS3)	32	R/W	0000_0000h	68.6.15/ 2710
C3C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR3)	32	R/W	0000_0000h	68.6.16/ 2711
C40	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS4)	32	R/W	0000_0000h	68.6.15/ 2710
C44	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR4)	32	R/W	0000_0000h	68.6.16/ 2711
C48	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS5)	32	R/W	0000_0000h	68.6.15/ 2710
C4C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR5)	32	R/W	0000_0000h	68.6.16/ 2711
C50	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS6)	32	R/W	0000_0000h	68.6.15/ 2710
C54	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR6)	32	R/W	0000_0000h	68.6.16/ 2711
C58	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS7)	32	R/W	0000_0000h	68.6.15/ 2710
C5C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR7)	32	R/W	0000_0000h	68.6.16/ 2711
C60	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS8)	32	R/W	0000_0000h	68.6.15/ 2710

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C64	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR8)	32	R/W	0000_0000h	68.6.16/ 2711
C68	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS9)	32	R/W	0000_0000h	68.6.15/ 2710
C6C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR9)	32	R/W	0000_0000h	68.6.16/ 2711
C70	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS10)	32	R/W	0000_0000h	68.6.15/ 2710
C74	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR10)	32	R/W	0000_0000h	68.6.16/ 2711
C78	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS11)	32	R/W	0000_0000h	68.6.15/ 2710
C7C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR11)	32	R/W	0000_0000h	68.6.16/ 2711
C80	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS12)	32	R/W	0000_0000h	68.6.15/ 2710
C84	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR12)	32	R/W	0000_0000h	68.6.16/ 2711
C88	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS13)	32	R/W	0000_0000h	68.6.15/ 2710
C8C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR13)	32	R/W	0000_0000h	68.6.16/ 2711
C90	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS14)	32	R/W	0000_0000h	68.6.15/ 2710
C94	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR14)	32	R/W	0000_0000h	68.6.16/ 2711
C98	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS15)	32	R/W	0000_0000h	68.6.15/ 2710
C9C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR15)	32	R/W	0000_0000h	68.6.16/ 2711
CA0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS16)	32	R/W	0000_0000h	68.6.15/ 2710
CA4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR16)	32	R/W	0000_0000h	68.6.16/ 2711
CA8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS17)	32	R/W	0000_0000h	68.6.15/ 2710
CAC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR17)	32	R/W	0000_0000h	68.6.16/ 2711
CB0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS18)	32	R/W	0000_0000h	68.6.15/ 2710
CB4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR18)	32	R/W	0000_0000h	68.6.16/ 2711
CB8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS19)	32	R/W	0000_0000h	68.6.15/ 2710

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CBC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR19)	32	R/W	0000_0000h	68.6.16/ 2711
CC0	Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)	32	R/W	0000_0000h	68.6.17/ 2711
CC4	Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)	32	R/W	0000_0000h	68.6.18/ 2712
CC8	Flash memory concurrent overflow register (MEMU_FLASH_OFLW0)	32	w1c	0000_0000h	68.6.19/ 2712

68.6.2 Control register (MEMU_CTRL)

Access: User read/write

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SWR	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_CTRL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SWR	Software Reset bit. Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.
17–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

68.6.3 Error flag register (MEMU_ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO,F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a '1'. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Access: User read/write

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											PR_CE	PR_UCE	PR_CEO	PR_UCO	PR_EBO
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			F_CE	F_UCE	F_CEO	F_UCO	F_EBO	0			SR_CE	SR_UCE	SR_CEO	SR_UCO	SR_EBO
W				w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_ERR_FLAG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: • a new Flash ECC uncorrectable error was detected. • a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SR_CE	System RAM ECCand MBIST Correctable Error detect flag. Indicates that a new and unique System RAM ECCand MBISTcorrectable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
28 SR_UCE	System RAM ECCand MBIST Uncorrectable Error Detect flag. Asserted when: • A new System RAM ECCand MBIST uncorrectable error is detected • A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
29 SR_CEO	System RAM ECCand MBIST Correctable error Overflow flag. Asserted when the System RAM ECCand MBIST correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
30 SR_UCO	System RAM ECCand MBIST Uncorrectable error Overflow flag. Asserted when the System RAM ECCand MBIST uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECCand MBIST Error buffer Overflow.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	Flag asserted when the internal error processing buffer of System RAM ECCand MBIST MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

68.6.4 Debug register (MEMU_DEBUG)

This register is used to check the connections between MEMU & FCCU.

The error output of the MEMU towards FCCU shall stay set if forced until reset by SW via the MEMU in the normal way.

Access: User read/write

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											FR_PR_CE	FR_PR_UCE	FR_PR_CEO	FR_PR_UCO	FR_PR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			FR_F_CE	FR_F_UCE	FR_F_CEO	FR_F_UCO	FR_F_EBO	0			FR_SR_CE	FR_SR_UCE	FR_SR_CEO	FR_SR_UCO	FR_SR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_DEBUG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag going towards FCCU to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag going towards FCCU to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag going towards FCCU to 1.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag going towards FCCU to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag going towards FCCU to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag going towards FCCU to 1.
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag going towards FCCU to 1.
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag going towards FCCU to 1.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag going towards FCCU to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag going towards FCCU to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag going towards FCCU to 1.

68.6.5 System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 20h offset + (8d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF.

Table continues on the next page...

MEMU_SYS_RAM_CERR_STS*n* field descriptions (continued)

Field	Description
	Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

68.6.6 System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 24h offset + (8d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

68.6.7 System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W		▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

68.6.8 System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (SYS_RAM_UNCERR_STS[VLD]) = 0.

68.6.9 System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 78h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

68.6.10 Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 620h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

68.6.11 Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 624h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

68.6.12 Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 630h offset = 630h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_PERIPH_RAM_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

68.6.13 Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 634h offset = 634h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (PERIPH_RAM_UNCERR_STS[VLD]) = 0.

68.6.14 Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 638h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

68.6.15 Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + C20h offset + (8d × i), where i=0d to 19d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R					0								BAD_BIT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid . Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in flash memory where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

68.6.16 Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + C24h offset + (8d × i), where i=0d to 19d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

68.6.17 Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC0h offset = CC0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_FLASH_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

68.6.18 Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC4h offset = CC4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (FLASH_UNCERR_STS[VLD]) = 0.

68.6.19 Flash memory concurrent overflow register (MEMU_FLASH_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC8h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

68.7 Functional description

In this section, the term "bad bit" and "syndrome" are equivalent, and are used to indicate the position of the bit in error.

68.7.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known errors which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit ='1' for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

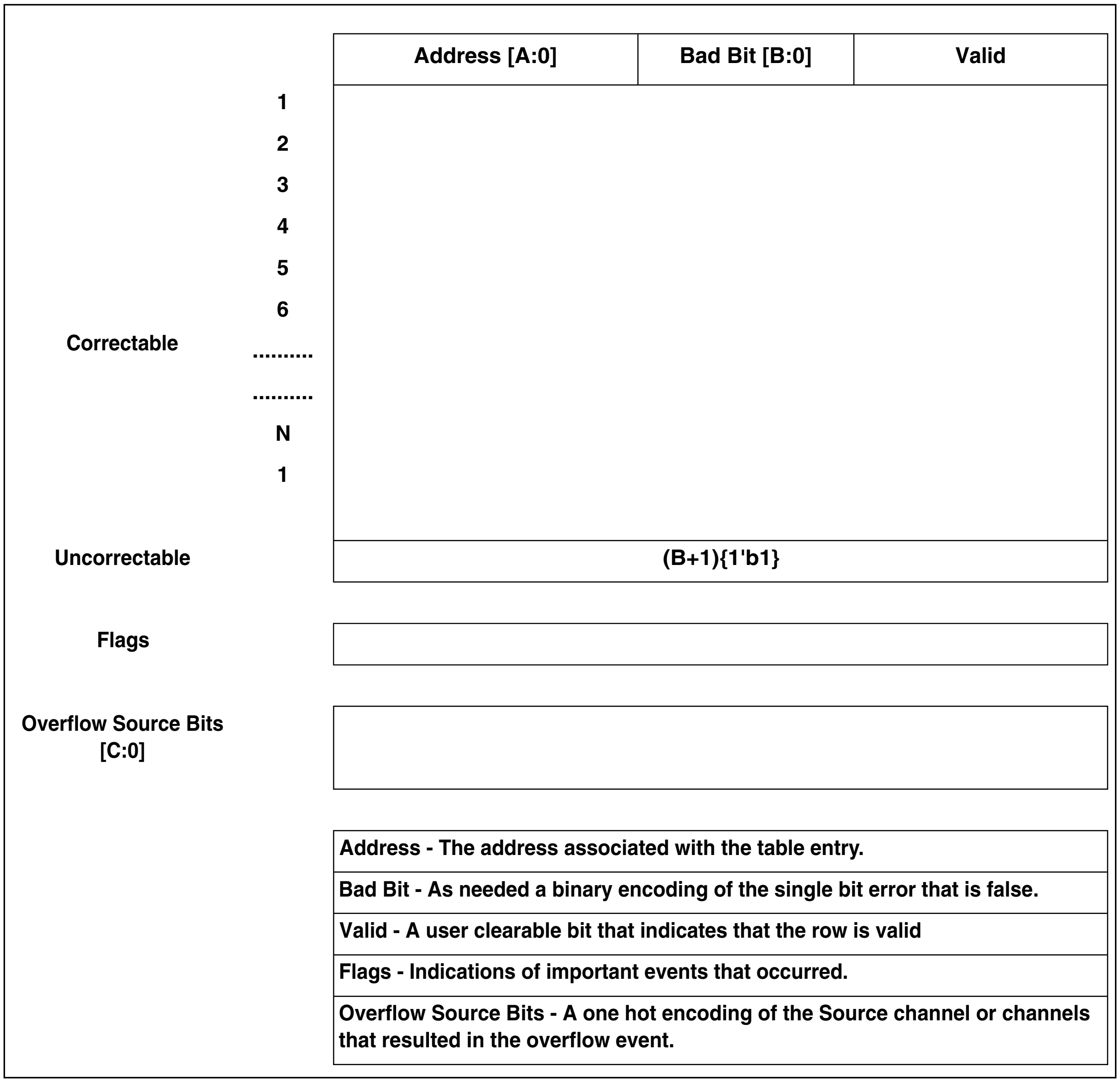
The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address. The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows:

1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any error handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU writes an error with the same address that Software wants to program, both the errors will be stored in the reporting table (i.e. duplicate entries).

68.7.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.



Address - The address associated with the table entry.

Bad Bit - As needed a binary encoding of the single bit error that is false.

Valid - A user clearable bit that indicates that the row is valid

Flags - Indications of important events that occurred.

Overflow Source Bits - A one hot encoding of the Source channel or channels that resulted in the overflow event.

Figure 68-4. Generic representation of the reporting table of one MEMU reporting block

68.7.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

1. Error buffer overflow–More than two memories reporting an error at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
   - More than 2 errors detected at same time. This would lead to overflow in the error buffer. MEMU can process only one error at a time while storing the other in the error buffer. So this would be indicated as an overflow. The uniqueness check in the input buffer does not "guarantee" that an ECC-supervised memory with only one error will not be marked as overflow source. If that one error occurs together with two (or more) additional errors at the same cycle it can get flagged as an overflow.
2. Correctable/Uncorrectable Error Overflow–Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
   - An ECC error which matches the address of an MBIST error is dropped (same as if it matched the address of an ECC error). In case, where a MBIST error comes with ECC error already in the table, both the address and bad bits are compared to determine uniqueness.
3. The Overflows will be stored in the bit-wise order i.e., if the error bit 31 reports an error, the bit 31 of the overflow register will be set to '1'.

Chapter 69
Fault Collection and Control Unit (FCCU)

69.1 Introduction

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect faults and to place the device into a safe state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

The FCCU offers a systematic approach to fault collection and control. The distinctive features of the module are:

- Redundant collection of hardware checker (for example, RCCU) results
- Redundant collection of fault information from safety relevant modules on the device
- Collection of test results
- Configurable fault control
- Configurable internal reactions for each NCF:
  - No reaction
  - IRQ
  - Short functional reset
  - Long functional reset
  - NMI
- External reactions via configurable output pins
- Watchdog timer for the reconfiguration phase

- The FCCU's configuration is lockable:
  - Permanently locked until next reset or transiently locked until a specific key is written. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register.
- One of the error out pins is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the error out protocol is configured to be a toggling protocol, for example, dual rail and time switching.
- After power on, the EOUT signals have high impedance.[1] They show operational state only on software request.
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in Bi-stable mode only).
- The overall fault detection, processing and indication time is less than 10 ms.
- The actual maximum time is 5 CLKSAFE cycles**.**

The FCCU circuitry is checked at the start-up by the self-checking procedure. The FCCU is operative with the default configuration immediately after the completion of the self-checking procedure. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable. The default configuration can be modified only in the CONFIG state. FCCU is designed to function when CLKSYS is faster than the CLKSAFE clocks.

69.1.1 Acronyms and abbreviations

Term	Description
CF	Critical fault
CPU	Central processing unit
EOUT	Error out
FOSU	FCCU output supervision unit
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRQ	Interrupt request
MC_ME	Mode entry module

Table continues on the next page...

[1] Actual value depends on device specific setting at pad level.

Term	Description
NCF	Noncritical fault
NMI	Nonmaskable interrupt
RCC	Redundancy control checkers
RCCU	Redundancy control checker unit
SoC	System-on-chip
STCU	Self-test control unit
SW	Software
WKPU	Wake-up unit

69.2 Main features

- Management of non-critical faults
- HW or SW fault recovery management
- Fault detection and collection
- Fault injection (fake faults)
- External reaction (fault state): EOUT signaling. Error indication via the pin(s) is controlled by the FCCU.
- Internal chip reactions (alarm state): interrupt request
- Internal chip reactions (fault state):
  - long functional reset request pulse
  - short functional reset request pulse
  - NMI
- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Internal (to the FCCU) watchdog timer for the reconfiguration phase
- Configuration lock

69.3 Block diagram

The following figure is a top-level diagram of the FCCU module.

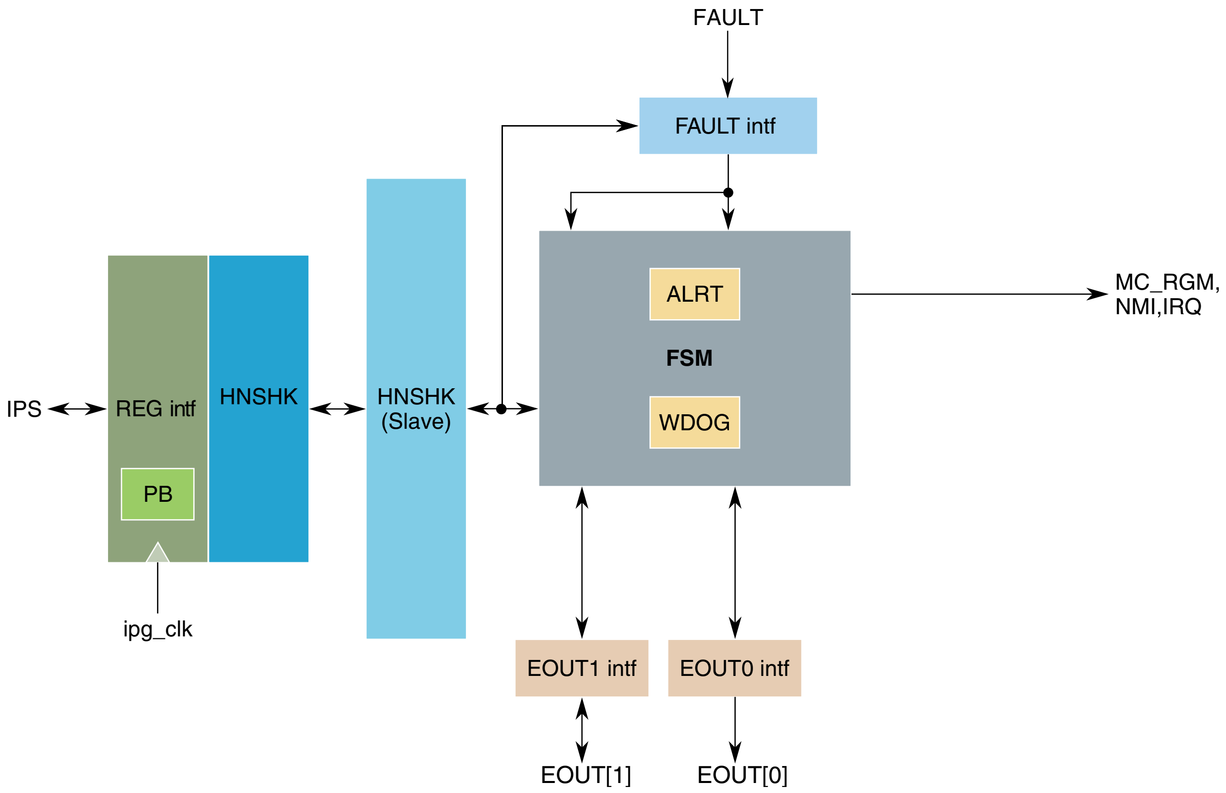


Figure 69-1. FCCU block diagram

This table describes the FCCU submodules.

Table 69-1. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSM's ability to support the handshake between the REG interface and the FSM unit due to the usage of 2 asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: • Watchdog timer (WDG) • Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

69.4 Signal description

69.4.1 Signals

This table describes the signals on the boundary of FCCU.

Signal	Direction	Function	Active level	Clock
Clock and reset				
CLKSYS	Input	System Clock—Supplies the primary clock.	Rising edge	—
CLKSAFE0	Input	Safe Clock 0—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE1 are often referred to as the same clock (CLKSAFE).	Rising edge	—
CLKSAFE1	Input	Safe Clock 1—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE0 are often referred to as the same clock (CLKSAFE).	Rising edge	—
EOUT interface				
EOUT[1:0]	Output	Error Output—Indicate to off-chip logic that a fault has occurred. For more information, see EOUT interface.	Programmable	CLKSAFE0, CLKSAFE1
WKPU interface				
NMIWKPU	Input	WKPU Nonmaskable Interrupt Request—Receives nonmaskable interrupt requests from the WKPU module.	Low	CLKSYS
WAKEWKPU	Output	Wake WKPU—Wakes the WKPU module to bring the chip out of dormant state when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	High	CLKSAFE0, CLKSAFE1
Interrupt interface				
NMIOUT	Output	Nonmaskable Interrupt Output—Sends a nonmaskable interrupt request to the processor core or cores when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	Low	CLKSYS

69.5 Functional description

69.5.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is deasserted. That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line stays at 0. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched external to the FCCU in the module where the fault occurred. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the fault is recoverable by itself when the fault condition gets corrected.

This table describes the types of resets.

Reset type	Function
Destructive	Initializes the entire chip as the result of a power-failure condition.
Long functional	Initializes the digital circuitry except for FCCU and STCU2.
Short functional	Initializes the digital circuitry except for OCOTP, FCCU, and STCU2.

For more information on each type of reset, see the MC_RGM and reset chapters.

69.5.2 FSM description

FCCU functionality is depicted by the finite state machine (FSM) state diagram.

Basically four states are identified with the following meaning:

- Configuration: Used only to modify the configuration of FCCU from its default. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, noncritical fault masking) can be accessed in write mode only in Configuration state.

  Configuration state is accessible only in Normal state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset which also resets the FCCU., The transient lock register is unlocked by writing BCh into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than BCh). Key to lock FCCU permanently is FFh, written in the FCCU_PERMNT_LOCK register.

  After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked.

  The locking feature only restricts the FSM movement into Configuration state. Once the user enters Configuration state and then tries to lock the configuration by writing either to TRANS_LOCK or PERMT_LOCK, the locking of configuration will be effective after FCCU moves to Normal state, it will not be effective in the current Configuration state.

  The Configuration to Normal state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to Normal appears at the same time, watchdog timeout has the priority and hence the Configuration registers (those that are writable only in Configuration state) are reset to their default values. The movement to Normal is made.

  The incoming faults, occurring during the configuration phase (Configuration state) are latched in order to process them when FCCU is moved into Normal state, according to the new configuration.

  All pending faults that occur during Configuration state result in both of the following:

  - Highest-priority state transition
  - Interrupt generation (NMI or alarm IRQ)

  If the state transition occurs, it gives the reset reaction corresponding to the worst case based on all the faults (pending or nonpending faults) that occurred during Configuration state.

- Normal: This is FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following state transitions occur on one of the following events:

    - unmasked noncritical faults with the timeout disabled → FCCU moves to Fault state
    - unmasked noncritical faults with the timeout enabled → FCCU moves to Alarm state
    - masked noncritical faults → FCCU stays in Normal state
- Alarm: FCCU moves into Alarm state when an unmasked noncritical fault occurs and the timeout is enabled. The transition to Alarm state goes along with an interrupt alarm, if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to Fault state. The timeout is reinitialized if FCCU enters Normal state. The timeout restarts following the recovery from Fault state.
- Fault: FCCU moves into Fault state when one of the following condition occurs:
    - timeout related to a noncritical fault when FCCU is in Alarm state
    - unmasked noncritical faults with the timeout disabled

The transition from Normal/Alarm to Fault state goes along with the generation of:

- NMI interrupt (optional)
- EOUT signaling (optional)
- SW option: Soft reaction (Short functional reset request pulse if configured)
- SW option: Hard reaction (Long functional reset request pulse if configured)
- Non Maskable Interrupt (NMI) is routed only to the Safety Core

After moving to Fault state, if there is either a previous pending fault or a new fault for which NMI is enabled, NMI generation takes place.

Multiple faults can occur at the same time. Consider the case where there is a fault for which Alarm state is enabled and IRQ is programmed along with a fault for which Alarm state is disabled. In that case, the FSM moves to Fault state, but with the generation of alarm IRQ for the fault which has IRQ enabled.

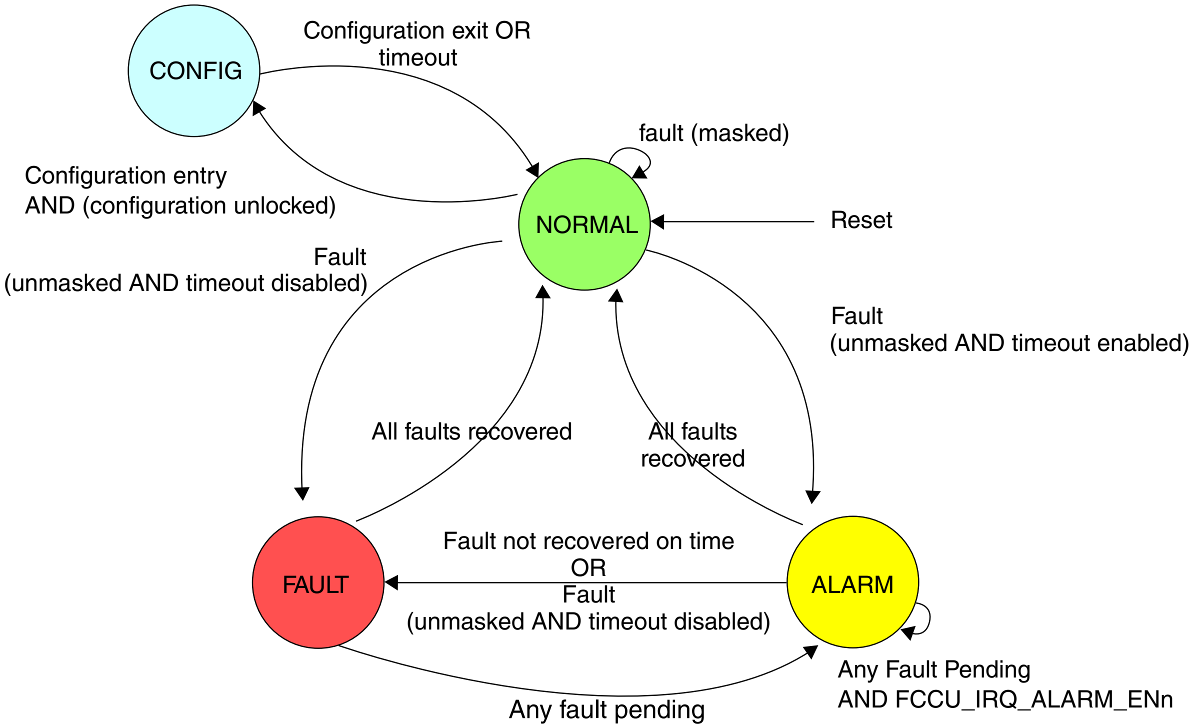

CONFIG
Configuration exit OR timeout
Configuration entry AND (configuration unlocked)
fault (masked)
NORMAL
Reset
Fault (unmasked AND timeout disabled)
Fault (unmasked AND timeout enabled)
All faults recovered
All faults recovered
FAULT
Fault not recovered on time OR Fault (unmasked AND timeout disabled)
ALARM
Any Fault Pending AND FCCU_IRQ_ALARM_ENn
Any fault pending

Figure 69-2. FCCU state diagram

69.5.3 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (non-critical) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a non-critical fault (unmasked and with time-out disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the non-critical faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the non-critical faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will move to the ALARM state (if any non-critical fault is still pending and the time-out is not elapsed).

In general, no fault nesting is supported except for the non-critical faults that cause an ALARM to FAULT state transition. In this case the NCF timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

69.5.4 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a non-critical FAULT management (ALARM state), see Figure 69-3 and Figure 69-4, is as follows:

- non-critical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- system state: RUN
- alarm interrupt management: FAULT recovery (by SW)
  - FCCU state transition ALARM → NORMAL

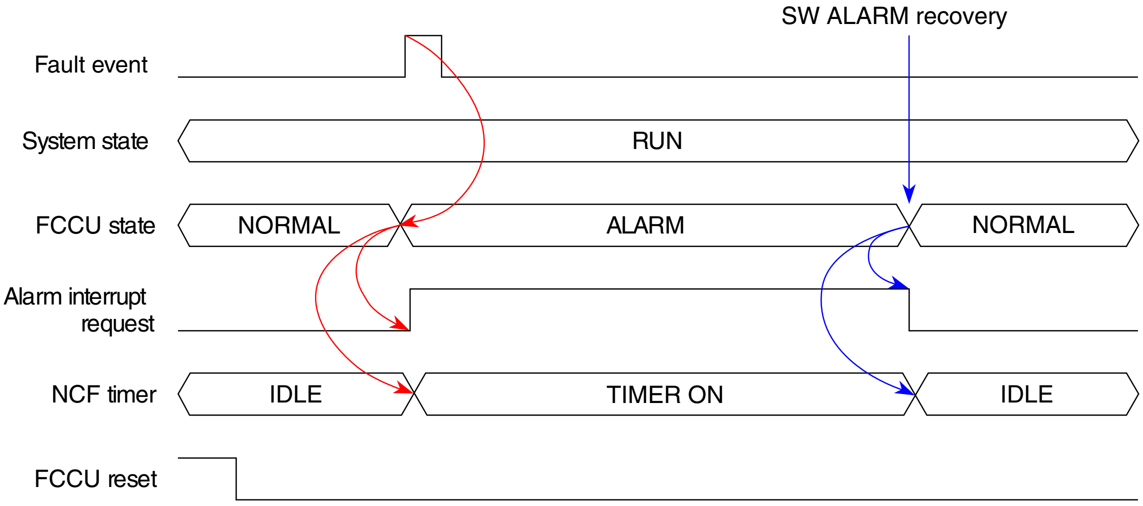


Figure 69-3. Non-critical FAULT (ALARM state) recovery (a)

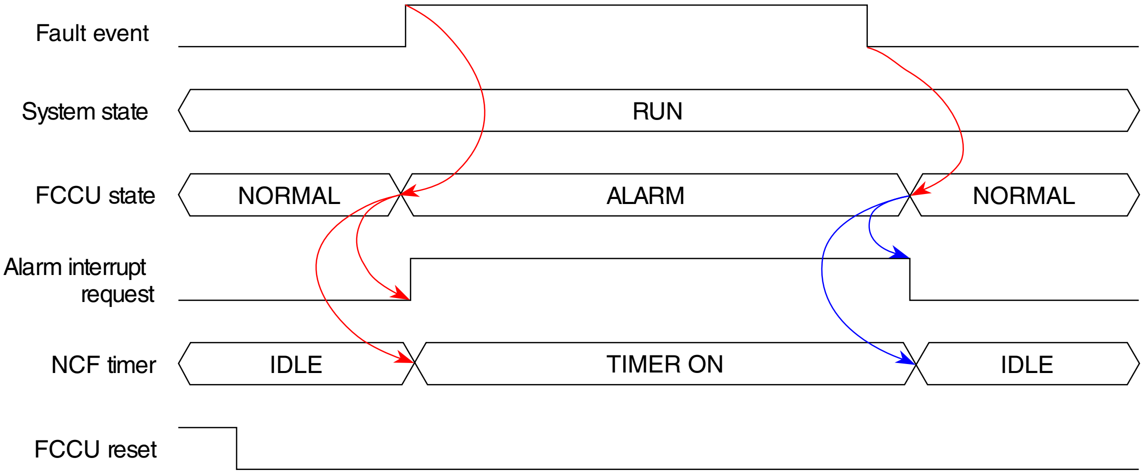


Figure 69-4. Non-critical FAULT (ALARM state) recovery (b)

A typical sequence related to a FAULT management (ALARM → FAULT state), see Figure 69-5, is as follows:

- non-critical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
    - alarm interrupt request (if enabled)
    - time-out running
- FCCU state transition (following the time-out trigger): ALARM → FAULT
    - NMI assertion (if enabled)
- system state transition: RUN → SAFE
- NMI interrupt management (if enabled)
    - FAULT recovery (by SW): FCCU state transition FAULT → NORMAL
    - system state transition: SAFE → RUN

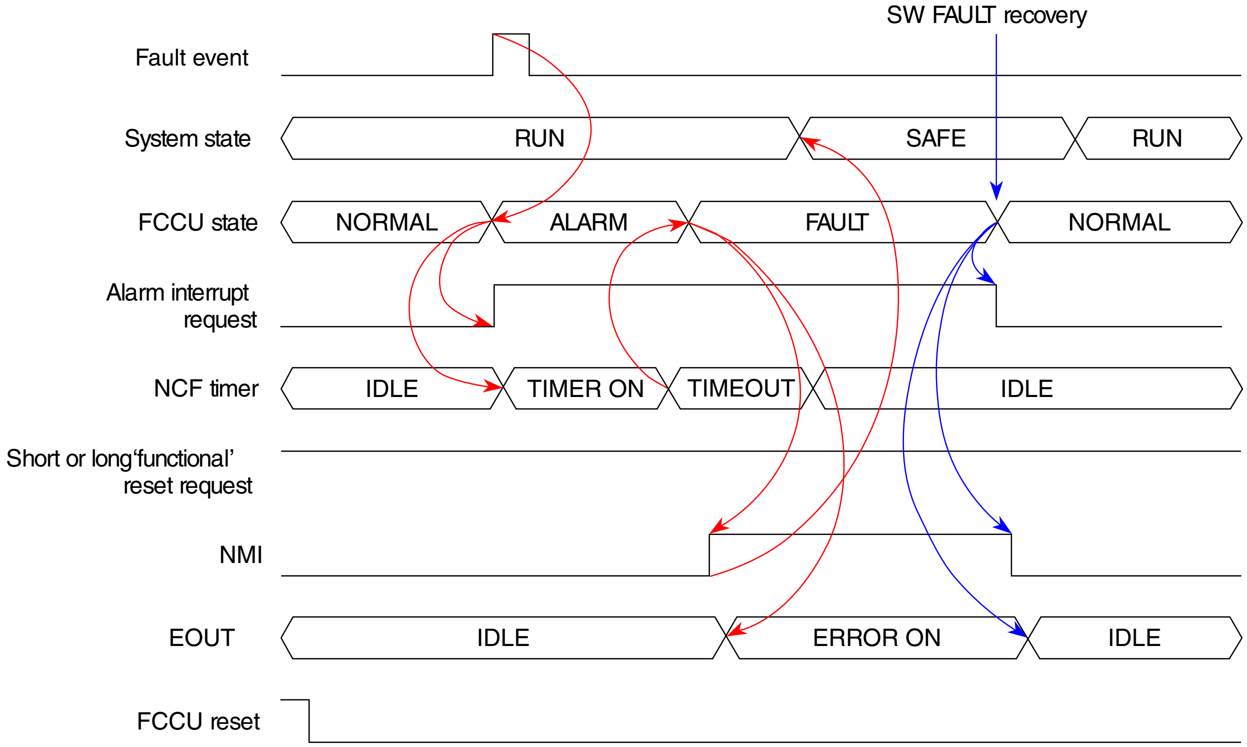


Figure 69-5. FAULT (ALARM → FAULT state) recovery

69.5.5 NMI/WKPU interface

NMIOUT is connected to all processor cores on the chip but is masked when any one of the following is true:

- An FCCU asynchronous reset is in progress.
- The chip is in RESET mode.
- The chip has automatically entered DRUN mode from RESET mode and software has not yet requested another mode.

This figure shows the logical scheme.

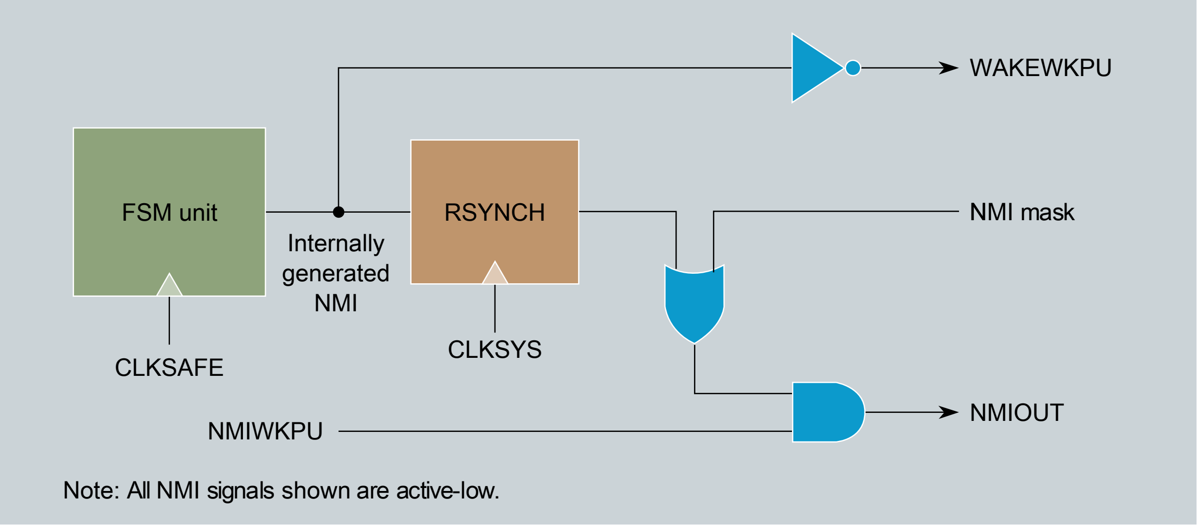


Figure 69-6. NMI/WKPU scheme

69.5.6 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the FCCU.

  The STCU fault signals are processed by the FCCU when the SoC is re-booted following the self-testing procedure. The STCU includes a status register that stores the self-testing results (flags).
- a mask that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self-testing procedure, depending on the STCU results, three cases are applicable:

1. STCU completes the self-testing procedure successfully. The SoC reboots and the FCCU is responsible for fault collection and indication.
2. STCU completes the self-testing procedure with low severity failures. The FCCU records the fault condition but does not, by default, trigger a reset cycle.
3. STCU completes the self-testing procedure with serious failures. The FCCU triggers a reset cycle in response to this fault input from the STCU. Because the fault condition survives reset, new reset cycles are subsequently triggered until the reset escalation feature takes control, freezing the chip in the reset state until a POR. This feature is optionally programmable using MC_RGM.

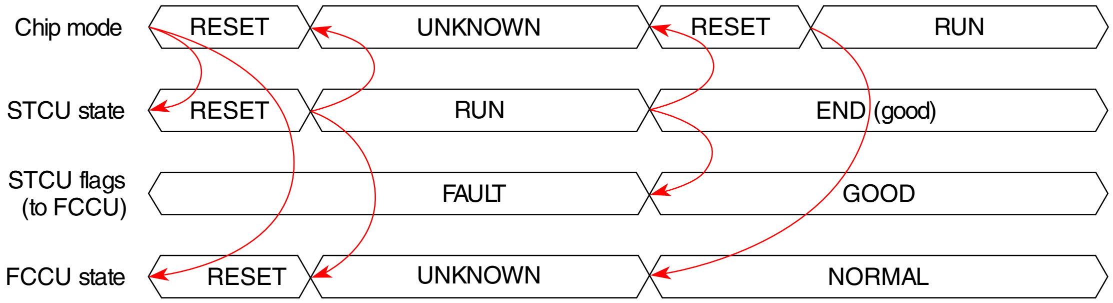


Figure 69-7. STCU-FCCU (case 1)

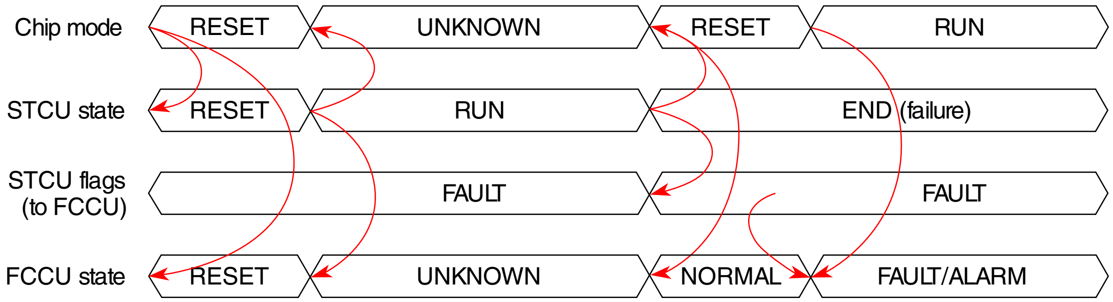


Figure 69-8. STCU-FCCU (case 2)

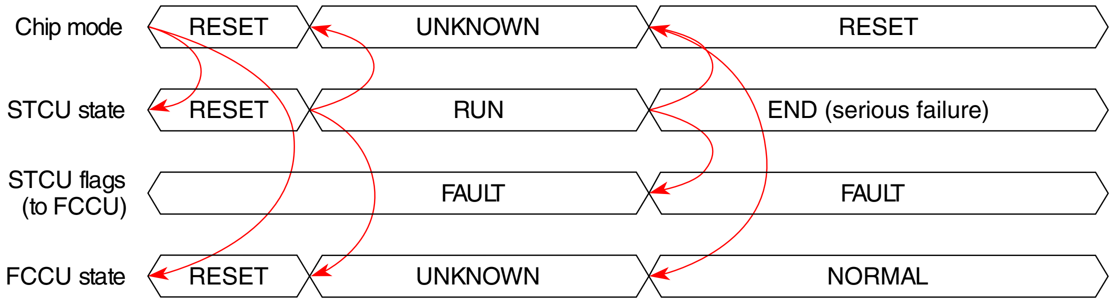


Figure 69-9. STCU-FCCU (case 3)

69.5.7 EOUT interface

The FCCU provides up to two bidirectional signals (EOUT[1:0]) as a failure indication to the external world. For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information.

Different fault-output modes (protocols) for the fault-output (EOUT) interface are supported (FCCU_CFG[FOM]):

- Dual-Rail
- Time-Switching

- Bistable
- Test

You can further configure the fault-output modes using the following attributes:

Attribute	Field	Setting used in the example diagrams and tables that follow
Switching mode	FCCU_CFG[SM]	Slow
Polarity selection	FCCU_CFG[PS]	For the faulty indication, EOUT1 is high, and EOUT0 is low.
Derived prescaler	FCCU_CFG[FOPE], FCCU_CFG[FOP]	0

With a CLKSAFE frequency of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and CLKSAFE (which detects the edge transition of the EOUT protocol) in Dual-Rail or Time-Switching fault-output mode.

In case of a failure event or on software request for EOUT indication, the signal(s) are set to the faulty state for a minimum time T_min, even if software tries to release it before. If software configures the error pins to OK(1), and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and the error signals indicate NOK, such as an incoming fault is not masked when software has set the error signal to high. During the T_min by a non-software fault, the FCCU FSM moves independently of this signal state (low), and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in, and it is not possible to set the signals to OK by software moving FCCU to Configuration state, as long as this timer is running. No software intervention is needed to bring the signal from the low state.

Software can bring the pin back to OK state by clearing the faults and waiting for the T_min interval to expire, after which the FCCU automatically enters Normal state and the error signal indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

These resets influence the state of the failure indication pins:

- Power on reset
- Destructive reset (including when an FOSU timeout occurs due to FCCU failing to react)
- Chip pin reset (RESET_B)

Other resets should not influence the state of the failure indication pins.

NOTE

The transition from Fault to Configuration state is not possible but is shown to display the behavior in all four states—reset, normal, error, and configuration.

NOTE

FCCU remains in Normal state when a fault is disabled; therefore, <i>NA</i> (not applicable) appears in the following table.

NOTE

The following table is valid only after software has enabled the EOUT signals (FCCU_CFG[FCCU_SET_AFTER_RESET].

Table 69-2. FCCU error pin behavior on state transitions

<table>
<tr><th rowspan="3">Fault signaling state</th><th colspan="3">Transition from Normal to Fault state</th><th colspan="3" rowspan="2">Normal+Configuration states</th><th colspan="3">Transition from Normal to Alarm state</th></tr>
<tr><th colspan="3">with Fault disabled | enabled</th><th colspan="3">with Fault disabled | enabled</th></tr>
<tr><th>Internal error pin</th><th>Error pin enable</th><th>Error pad</th><th>Internal error pin</th><th>Error pin enable</th><th>Error pad</th><th>Internal error pin</th><th>Error pin enable</th><th>Error pad</th></tr>
<tr><td>Disabled</td><td>NA | OK</td><td>NA | 1</td><td>NA | OK</td><td>OK</td><td>1</td><td>OK</td><td>NA | OK</td><td>NA | 1</td><td>NA | OK</td></tr>
<tr><td>Enabled</td><td>NA | NOK</td><td>NA | 1</td><td>NA | NOK</td><td>OK</td><td>1</td><td>OK</td><td>NA | OK</td><td>NA | 1</td><td>NA | OK</td></tr>
</table>

NOTE

The first time FCCU is in Configuration state, the EOUT pads are in a high-impedance state; thereafter, their states reflect the programmed EOUT protocol.

69.5.7.1 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual-rail encoded circuits use two wires to carry each bit. The encoding scheme is given in Table 69-3 and the related timing diagram is given in Figure 69-10 and Figure 69-11.

Table 69-3. Dual-rail encoding

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	00	toggling
faulty	11	

Table continues on the next page...

Table 69-3. Dual-rail encoding (continued)

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
reset	Hi impedence[1]	no toggling
configuration	same as NORMAL	toggling

[1] Final value depends on the SoC setting at pad level.

As long as FCCU is in NORMAL or ALARM state, output will show "non-faulty" signal. Output pins EOUT[0:1] will toggle between 01 and 10 with a given frequency of 61 Hz.

In the RESET phase the output pins are set as "high impedance".

Note

> Figure 69-10 and Figure 69-11 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

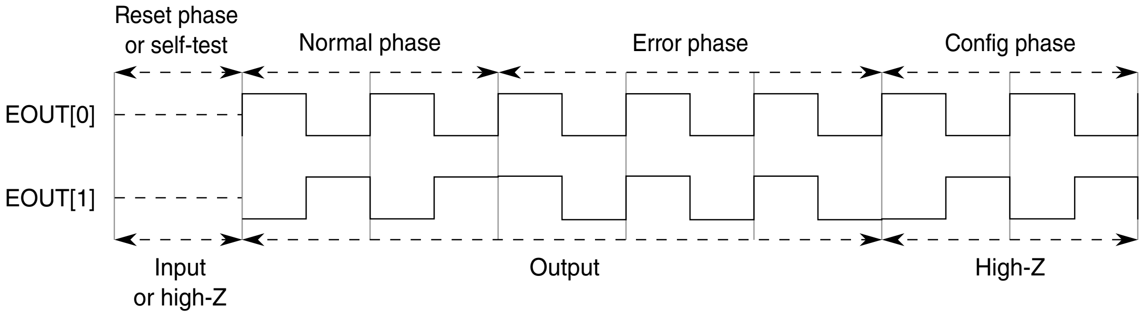


Figure 69-10. Dual-rail protocol (slow switching mode)

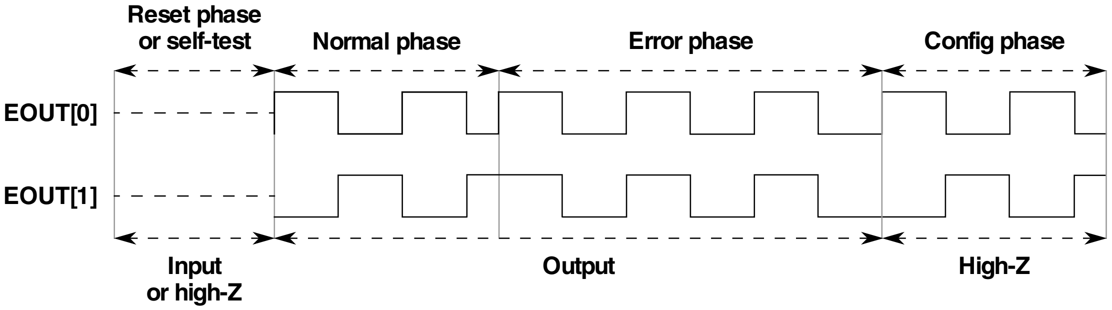


Figure 69-11. Dual-rail protocol (fast switching mode)

69.5.7.2 Time-switching protocol

The encoding scheme is given in Table 69-4 and the related timing diagram is given in Figure 69-12.

Table 69-4. Time-switching encoding

Logical state	Time-switching encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	toggling

[1] Final value depends on device specific settings at pad level.

As long as FCCU is in NORMAL or ALARM state, outputs will show "non-faulty" signal. Output pins #0, #1 will toggle between 01 and 10 with frequency specified by the CFG[FOP].

In the FAULT state, the output pin EOUT[0] is set as low.

In Time Switching mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[2]

Note

Figure 69-12 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

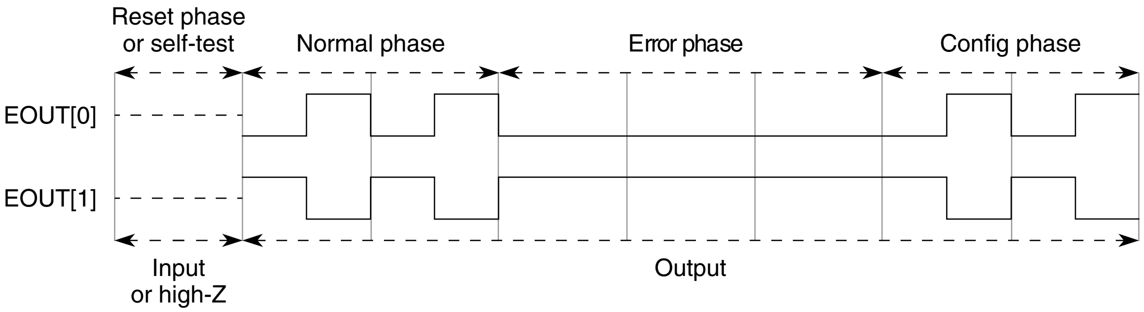


Figure 69-12. Time-switching protocol

[2] Actual value depends on device specific settings at pad level.

69.5.7.3 Bi-stable protocol

The encoding scheme is given in Table 69-5 and the related timing diagram is given in Figure 69-13.

Table 69-5. Bi-stable encoding

Logical state	Bi-stable encoding (output pins EOUT[1:0])	Note
non-faulty	01	no toggling
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	no toggling

[1] Final value depends on device specific settings at pad level.

In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, "no-faulty" state is indicated. In Bi-stable mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[3]

Note

> Figure 69-13 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

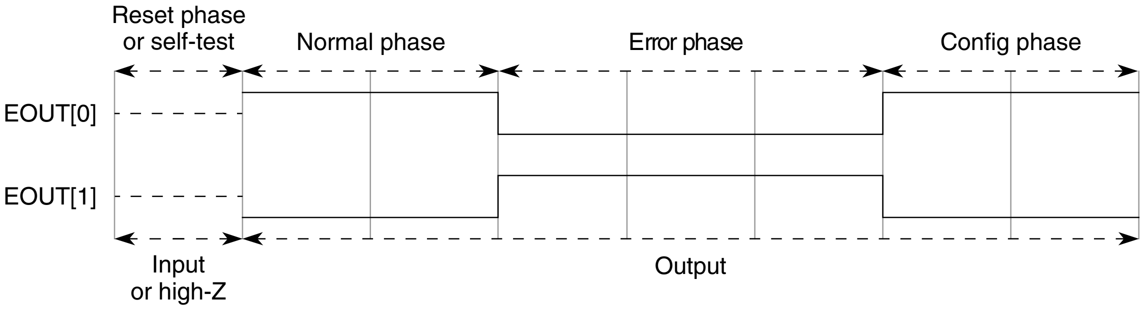


Figure 69-13. Bi-stable protocol

69.5.8 Error signal flow

See the chip-specific FCCU information for the diagram of the signal flow.

[3] Actual value depends on device specific settings at pad level.

69.6 Register descriptions

The FCCU registers are listed in the table below. Any address offset not explicitly mentioned in this table is reserved.

The FCCU supports word (32-bit), half-word (16-bit), and byte (8-bit) read and write accesses.

Follow these register-access guidelines:

- Don't read from or write to any addresses that are not shown in the following table. Doing so may result in a transfer error.
- Don't write to any of the configuration registers unless FCCU is in Configuration state. Doing so results in a transfer error.
- Don't write to FCCU_TRANS_LOCK or FCCU_PERMNT_LOCK unless your code runs in Supervisor mode. Doing so results in a transfer error.

All the registers accessible in write mode only in CONFIG state are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible NCF failure source a different reaction shall be configurable through the use of NMI, IRQ, long/short reset selection registers as well as no reaction by disabling the former registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

FCCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control (FCCU_CTRL)	32	R/W	0000_00C0h	69.6.1/2738
4	Control Key (FCCU_CTRLK)	32	W	0000_0000h	69.6.2/2741
8	Configuration (FCCU_CFG)	32	R/W	See section	69.6.3/2742
1C	Noncritical Fault Configuration (FCCU_NCF_CFG0)	32	R/W	See section	69.6.4/2745
20	Noncritical Fault Configuration (FCCU_NCF_CFG1)	32	R/W	See section	69.6.4/2745
24	Noncritical Fault Configuration (FCCU_NCF_CFG2)	32	R/W	See section	69.6.4/2745

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4C	Noncritical Fault State Configuration (FCCU_NCFS_CFG0)	32	R/W	See section	69.6.5/2746
50	Noncritical Fault State Configuration (FCCU_NCFS_CFG1)	32	R/W	See section	69.6.5/2746
54	Noncritical Fault State Configuration (FCCU_NCFS_CFG2)	32	R/W	See section	69.6.5/2746
58	Noncritical Fault State Configuration (FCCU_NCFS_CFG3)	32	R/W	See section	69.6.5/2746
5C	Noncritical Fault State Configuration (FCCU_NCFS_CFG4)	32	R/W	See section	69.6.5/2746
80	Noncritical Fault Status (FCCU_NCF_S0)	32	R/W	0000_0000h	69.6.6/2747
84	Noncritical Fault Status (FCCU_NCF_S1)	32	R/W	0000_0000h	69.6.6/2747
88	Noncritical Fault Status (FCCU_NCF_S2)	32	R/W	0000_0000h	69.6.6/2747
90	Noncritical Fault Key (FCCU_NCFK)	32	W	0000_0000h	69.6.7/2749
94	Noncritical Fault Enable (FCCU_NCF_E0)	32	R/W	See section	69.6.8/2750
98	Noncritical Fault Enable (FCCU_NCF_E1)	32	R/W	See section	69.6.8/2750
9C	Noncritical Fault Enable (FCCU_NCF_E2)	32	R/W	See section	69.6.8/2750
A4	Noncritical Fault Timeout Enable (FCCU_NCF_TOE0)	32	R/W	See section	69.6.9/2751
A8	Noncritical Fault Timeout Enable (FCCU_NCF_TOE1)	32	R/W	See section	69.6.9/2751
AC	Noncritical Fault Timeout Enable (FCCU_NCF_TOE2)	32	R/W	See section	69.6.9/2751
B4	Noncritical Fault Timeout (FCCU_NCF_TO)	32	R/W	See section	69.6.10/ 2752
B8	Configuration-State Timeout (FCCU_CFG_TO)	32	R/W	See section	69.6.11/ 2752
BC	IO Control (FCCU_EINOUT)	32	R/W	0000_0000h	69.6.12/ 2753
C0	Status (FCCU_STAT)	32	R	0000_0010h	69.6.13/ 2755
C4	NA Freeze Status (FCCU_N2AF_STATUS)	32	R	0000_0000h	69.6.14/ 2757
C8	AF Freeze Status (FCCU_A2FF_STATUS)	32	R/W	0000_0000h	69.6.15/ 2758
CC	NF Freeze Status (FCCU_N2FF_STATUS)	32	R/W	0000_0000h	69.6.16/ 2759
D0	FA Freeze Status (FCCU_F2A_STATUS)	32	R	0000_0000h	69.6.17/ 2760
DC	Noncritical Fault Fake (FCCU_NCFF)	32	R/W	0000_0000h	69.6.18/ 2761
E0	IRQ Status (FCCU_IRQ_STAT)	32	R/W	0000_0000h	69.6.19/ 2762
E4	IRQ Enable (FCCU_IRQ_EN)	32	R/W	0000_0000h	69.6.20/ 2763
E8	XTMR (FCCU_XTMR)	32	R	0000_0000h	69.6.21/ 2764
EC	Mode Controller Status (FCCU_MCS)	32	R	See section	69.6.22/ 2765
F0	Transient Lock (FCCU_TRANS_LOCK)	32	W	0000_0000h	69.6.23/ 2767

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F4	Permanent Lock (FCCU_PERMNT_LOCK)	32	W	0000_0000h	69.6.24/ 2768
F8	Delta T (FCCU_DELTA_T)	32	R/W	0000_0000h	69.6.25/ 2768
FC	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN0)	32	R/W	0000_0000h	69.6.26/ 2769
100	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN1)	32	R/W	0000_0000h	69.6.26/ 2769
104	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN2)	32	R/W	0000_0000h	69.6.26/ 2769
10C	NMI Enable (FCCU_NMI_EN0)	32	R/W	0000_0000h	69.6.27/ 2770
110	NMI Enable (FCCU_NMI_EN1)	32	R/W	0000_0000h	69.6.27/ 2770
114	NMI Enable (FCCU_NMI_EN2)	32	R/W	0000_0000h	69.6.27/ 2770
11C	Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN0)	32	R/W	0000_0000h	69.6.28/ 2771
120	Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN1)	32	R/W	0000_0000h	69.6.28/ 2771
124	Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN2)	32	R/W	0000_0000h	69.6.28/ 2771

69.6.1 Control (FCCU_CTRL)

The FCCU_CTRL register allows the following operations to be executed:

- moving the FCCU state from the NORMAL state into the CONFIG state
- moving the FCCU state from the CONFIG state into the NORMAL state
- reading or to clear the NCF status register
- reading the FCCU FSM status register
- reading or to clear the FCCU freeze registers
- reading the ALARM timer
- reading the Watchdog timer

Some critical operations require a key as defined in the FCCU_CTRLK register.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FILTER_BYPASS	FILTER_ WIDTH		0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						DEBUG	0	OPS		0	OPR				
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

FCCU_CTRL field descriptions

Field	Description
0 FILTER_ BYPASS	Filter bypass 0 glitch filter not bypassed 1 glitch filter bypassed
1–2 FILTER_WIDTH	Filter width 00 filters glitches up to 50 μs 01 filters glitches up to 75 μs 10 filters glitches up to 100 μs 11 filters glitches up to 100 μs
3–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUG	Debug Mode Enable Specifies whether Debug mode is enabled. If so, FCCU enters Debug mode when the Debug signal is asserted. When FCCU enters Debug mode, it halts operation and remains in the state it was in before it entered Debug mode. **NOTE:** FOSU doesn't halt when FCCU enters Debug mode. Therefore, FOSU can still cause a reset if a fault occurs while FCCU is in Debug mode.

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 OPS	Operation status. This bit can be read and cleared (via OP15 operation) by the software. 00 Idle 01 In progress 10 Aborted 11 Successful
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 OPR	Operation run. The SW application must not modify the OPR field (any write operation will be ignored) until the completion of the operation. Once the operation has been completed, the operation status (OPS) is set and the OPR field is automatically cleared (OPR = 000). The opcode OP12 must not be programmed. The OPR field is automatically set to OP12 when the FCCU_NCF_S registers are cleared by a write-clear operation into the related register. The opcode OP14 must not be programmed. The OPR field is automatically set to OP14 when the timeout occurs (FCCU_CFG_TO) during the configuration procedure => the FCCU state is automatically forced in NORMAL mode setting the default configuration. In this phase any write operation to the FCCU configuration registers is inhibited. In the case of OP15, the operation is successful immediately and OPS/OPR are cleared instantly. The operations OP21-OP30 are forbidden. In case of execution, they return an ABORT response without any side effect. The ABORT response occurs in the following cases: • wrong access (missing or wrong key) to the FCCU_NCF_S register (clear operation OP12) • wrong access (missing or wrong key) to the FCCU_CTRL register (OP1, OP2 operation) • OP1 (CONFIG command) execution when FCCU state /= NORMAL or configuration locked 00000 No operation [OP0] 00001 Set the FCCU into the CONFIG state [OP1] 00010 Set the FCCU into the NORMAL state [OP2] 00011 Read the FCCU state (see the FCCU_STAT register) [OP3] 00100 Read the FCCU frozen status flags (see the N2AF_STATUS register) [OP4] 00101 Read the FCCU frozen status flags (see the A2FF_STATUS register) [OP5] 00110 Read the FCCU frozen status flags (see the N2FF_STATUS register) [OP6] 00111 Read the FCCU frozen status flags (see the F2AF_STATUS register) [OP7] 01000 Reserved 01001 Reserved 01010 Read the NCF status register (see the FCCU_NCF_S register) [OP10] 01011 Reserved 01100 NCF status clear operation in progress (see the FCCU_NCF_S register) [OP12] 01101 Clear the freeze status registers (see the freeze registers) [OP13]

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	01110 CONFIG to NORMAL FCCU state (configuration timeout) in progress [OP14] 01111 Clear the operation status (OPS=Idle) [OP15] 10000 Reserved 10001 Read the ALARM timer (see the FCCU_XTMR register) [OP17] 10010 Reserved 10011 Read the CFG timer (see the FCCU_XTMR register) [OP19] 10100 Reserved 10101-11110 Forbidden 11111 Reserved

69.6.2 Control Key (FCCU_CTRLK)

The FCCU_CTRLK register implements the key access for the operations OP1, OP2 according to the following sequence:

1. Write the key into the FCCU_CTRLK register.
2. Write the FCCU_CTRL register (operations OP1 or OP2).

NOTE

The FCCU_CTRLK and FCCU_CTRL registers must be written with consecutive instructions. Both registers must be written as 32-bit values. Do not use read-modify-write instructions, such as bit field instructions, to modify these registers.

The FCCU_CTRLK register is not readable, a 0000_0000h value is always returned in case of read operation. The key must be written by a word (32 bits) data write operation.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	CTRLK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_CTRLK field descriptions

Field	Description
0–31 CTRLK	Control register key. = 913756AFh: Key for the operation OP1 = 825A132Bh: Key for the operation OP2

FCCU_CTRLK field descriptions (continued)

Field	Description

69.6.3 Configuration (FCCU_CFG)

Defines the global configuration for FCCU.

NOTE

This register is writable only when FCCU is in Configuration state.

NOTE

If you specify a new value for any of the fields in this register that affect the EOUT signals while the EOUT fault-indication timer is running (FCCU is indicating a fault on the EOUT signals), FCCU doesn't use the new settings you specified until after the EOUT fault-indication timer expires (FCCU stops indicating a fault on the EOUT signals).

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							FCCU_SET_AFTER_RESET	FCCU_SET_CLEAR		Reserved		Reserved			
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FOPE	Reserved		Reserved	Reserved	SM	PS	FOM			FOP					
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_CFG field descriptions

Field	Description
0–6 Reserved	This field is reserved.
7 FCCU_SET_ AFTER_RESET	This bit controls the enable of the EOUT signals after reset lifts. After power-on the EOUT signals shall be in a high-impedance state. **NOTE:** The actual value depends on the chip settings at the pad level. They will go to normal state only on software request, that is, this field is set to 1. It is software's responsibility to write 1 to this field; otherwise, the error pin stays in a high-impedance state after reset lifts and FCCU is not functioning. 0 FCCU's error indication stops functioning and the EOUT signals are in a high-impedance state (the actual value depends on the chip settings at the pad level). 1 Write to start FCCU functioning.
8–9 FCCU_SET_ CLEAR	Error pin state can be controlled by these bits These bits clear(0) and set(1) the error pin. Higher priority is of the FCCU_SET_AFTER_RESET bit's capability to lead the error pins to Hi-Z. **NOTE:** Actual value depends on the SoC settings at pad level. Switch to software configuration (01, 11) comes into effect when FCCU FSM leaves Configuration state. After this is in effect, software maintains priority over FCCU FSM except for the case of 11 mentioned above. The switch back to FSM control is again done by writing into these fields with 00 or 10. 00 FCCU acts independent of above software control 01 Both error signals are set to low. 10 FCCU acts independent of above software control. 11 Both error signals are set to high (write not possible when FCCU is already in the T_min timer phase or if FCCU enters Fault state by capture of a fault).
10–11 Reserved	This field is reserved.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
12–15 Reserved	This field is reserved. Always write the reset value to this field.
16 FOPE	Fault-Output (EOUT) Prescaler Extension For fault-output (EOUT) signaling, specifies the most significant bit of the derived prescaler that controls the signaling frequency. For more information, see the description of the FOP field.
17–18 Reserved	This field is reserved. Always write the reset value to this field.
19 Reserved	This field is reserved. Always write the reset value to this field.
20 Reserved	This field is reserved. Always write the reset value to this field.
21 SM	Fault-Output (EOUT) Switching Mode For fault-output (EOUT) signaling, controls how quickly the signals switch between faulty and nonfaulty indication. Applies only to Dual-Rail and Time-Switching fault-output modes. 0 Slow: No EOUT frequency violation during the FCCU state transition (Normal to Fault or vice versa, and Configuration to Normal). The indication transition occurs after a maximum delay equal to the duration of the semiperiod of the EOUT frequency. 1 Fast: The indication transition (Normal to Fault or vice versa, and Configuration to Normal) occurs immediately. A pulse with the minimum duration corresponding to the CLKSAFE period can occur in this mode. It implies a frequency violation of the EOUT protocol.
22 PS	Fault-Output (EOUT) Polarity Selection For fault-output (EOUT) signaling, controls the polarity of the signals for fault-output-mode indications that hold the signals low or high (versus toggling them or placing them in a high-impedance state). Applies only to Time-Switching fault-output mode (for faulty and configuration indications) and Bistable fault-output mode (for all indications). 0 For the faulty indication, EOUT1 is high, and EOUT0 is low. 1 For the faulty indication, EOUT1 is low, and EOUT0 is high.
23–25 FOM	Fault-Output (EOUT) Mode For fault-output (EOUT) signaling, controls the protocol of the signaling. **NOTE:** In the test modes, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs with CLKSAFE. 000 Dual-Rail (default state) [EOUT[1:0]= outputs] 001 Time-Switching [EOUT[1:0]= output to be used] 010 Bistable 011 Reserved 100 Reserved 101 Test 0 (controlled by the FCCU_EINOUT register; EOUT1 is an output; EOUT0 is an input) 110 Test 1 (controlled by the FCCU_EINOUT register; EOUT1 and EOUT0 are both outputs) 111 Test 2 (controlled by the FCCU_EINOUT register; EOUT1 is an input; EOUT0 is an output)
26–31 FOP	Fault-Output (EOUT) Prescaler For fault-output (EOUT) signaling, specifies the least significant bits of the derived prescaler that controls the signaling frequency. The derived prescaler is the concatenation of FOPE and FOP and is represented by *{FOPE, FOP}*.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	The following equation defines the signaling frequency in terms of the CLKSAFE frequency ($CLKSAFE_{freq}$) and the derived prescaler: $EOUT_{freq} = CLKSAFE_{freq} / ((\{FOPE, FOP\} + 1) \times 2 \times 2048)$ The following values and meanings are for the derived (7-bit) prescaler, not FOP alone. The meanings show the derived prescaler followed by the corresponding value by which the input clock frequency is then divided. 0000000: 0; divide by 2 × 2048 (4096) 0000001: 1; divide by 4 × 2048 (8192) 0000010: 2; divide by 6 × 2048 (12,288) … 1111101: 125; divide by 252 × 2048 (516,096) 1111110: 126; divide by 254 × 2048 (520,192) 1111111: 127; divide by 256 × 2048 (524,288) **NOTE:** The value of FOP after a Configuration-state timeout is 001000b (divide by 18 × 2048, or 36,864).

69.6.4 Noncritical Fault Configuration (FCCU_NCF_CFG*n*)

Contains the configuration of noncritical faults in terms of fault-recovery management. The configuration depends on the type of signaling of a fault event. Hardware-recoverable faults should be configured only if a previous latching stage captures and holds the physical fault; otherwise, the fault can be lost. All other faults should be configured as software faults.

NOTE

> These registers are writable only when FCCU is in Configuration state.

This table shows the noncritical-fault configuration-register channels.

Table 69-6. Noncritical-fault configuration-register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
1Ch	FCCU_NCF_CFG0	NCFC[31:0]
20h	FCCU_NCF_CFG1	NCFC[63:32]
24h	FCCU_NCF_CFG2	NCFC[95:64]

Address: 0h base + 1Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_CFG*n* field descriptions

Field	Description
0–31 NCFCx	Noncritical Fault Configuration. Defines the fault recovery mode. Hardware-recoverable faults are self-recovered (status flag clearing and related) if the root cause (input fault) has been removed. Software-recoverable faults are recovered (status flag clearing) by software clearing of the related status flag. See Table 69-6 for register offset to channel number relationship. 0 Hardware-recoverable fault 1 Software-recoverable fault

69.6.5 Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)

The FCCU_NCFS_CFGx registers contain the configuration of each non-critical fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.

Table 69-7 shows FCCU configuration register channels.

Table 69-7. FCCU NCFS configuration register channels

Address offset	Register name	Channel range (x)
4Ch	FCCU_NCFS_CFG0	NCFSCx[15:0]
50h	FCCU_NCFS_CFG1	NCFSCx[31:16]
54h	FCCU_NCFS_CFG2	NCFSCx[47:32]
58h	FCCU_NCFS_CFG3	NCFSCx[63:48]
5Ch	FCCU_NCFS_CFG4	NCFSCx[79:64]

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFSCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCFS_CFG*n* field descriptions

Field	Description
0–31 NCFSCx	Non-critical fault state configuration See Table 69-7 for register offset to channel number relationship. 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

69.6.6 Noncritical Fault Status (FCCU_NCF_S*n*)

The FCCU_NCF_Sx register contains the latched fault indication collected from the non-critical fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the NCF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering from the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_NCF_Sx register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_NCFK register.

2. Clear the status (flag) bit NCFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_NCF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root .

NOTE

There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_NCF_Sx read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_NCF_Sx register.

In case of re-configuration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_NCF_Sx must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following faults are ignored:

- to write a wrong key into the FCCU_NCFK register
- to attempt to clear a HW recoverable fault

Table 69-8 shows FCCU NCF status register channels.

Table 69-8. FCCU NCF status register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
80h	FCCU_NCF_S0	NCFS[31:0]
84h	FCCU_NCF_S1	NCFS[63:32]
88h	FCCU_NCF_S2	NCFS[95:64]

Address: 0h base + 80h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCFSx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCF_S*n* field descriptions

Field	Description
0–31 NCFSx	Non-critical fault status. The status bits related to the non-critical fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed. See Table 69-8 for register offset to channel number relationship. 0 No "non-critical" fault latched 1 "Non-critical" fault latched

69.6.7 Noncritical Fault Key (FCCU_NCFK)

The FCCU_NCFK register implements the key access to clear the status flags of the FCCU_NCF_Sx register.

The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx .

The key must be written for each FCCU_NCF_Sx clear operation.

The FCCU_NCFK register is not readable; a 0000_0000h value is always returned in case of read operation.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W	NCFK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFK field descriptions

Field	Description
0–31 NCFK	Non-critical fault key = AB34_98FEh

69.6.8 Noncritical Fault Enable (FCCU_NCF_E*n*)

The FCCU_NCF_En registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_NCF_Sn register is set (for debugging purposes), only the reaction is masked.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Any enabled fault should be programmed to result in a defined action. For example, set up an ALARM IRQ action by programming the IRQ Alarm Enable Register (FCCU_IRQ_ALARM_ENn).

Table 69-9 shows FCCU NCFE enable register channels.

Table 69-9. FCCU NCF enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
94h	FCCU_NCF_E0	NCFE[31:0]
98h	FCCU_NCF_E1	NCFE[63:32]
9Ch	FCCU_NCF_E2	NCFE[95:64]

Address: 0h base + 94h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_E*n* field descriptions

Field	Description
0–31 NCFEx	Non-critical fault enable See Table 69-9 for register offset to channel number relationship. 0 No actions following the respective non-critical fault assertion 1 FCCU moves to ALARM or FAULT state

69.6.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)

The FCCU_NCF_TOEx registers enable a transition from the NORMAL state into the ALARM state if the respective non-critical fault is enabled (NCFEx and NCFTOEx are set). In case the respective timeout is disabled (NCFTOEx is cleared) and the non-critical fault is enabled (NCFEx is set) the FCCU moves into the FAULT state if the related non-critical fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

NOTE

> These registers are writable only when the FCCU is in the CONFIG state.

Table 69-10 shows FCCU NCF timeout enable register channels.

Table 69-10. FCCU NCF timeout enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
A4h	FCCU_NCF_TOE0	NCFTOE[31:0]
A8h	FCCU_NCF_TOE1	NCFTOE[63:32]
ACh	FCCU_NCF_TOE2	NCFTOE[95:64]

Address: 0h base + A4h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFTOEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TOE*n* field descriptions

Field	Description
0–31 NCFTOEx	Fault timeout enable See Table 69-10 for register offset to channel number relationship. 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled

69.6.10 Noncritical Fault Timeout (FCCU_NCF_TO)

Defines the preset value of the timer for the recovery of enabled noncritical faults. Once FCCU enters Alarm state, following the assertion of a noncritical fault that is enabled (NCFEa and NCFTOEa are set), the timer starts the countdown.

If the fault is not recovered within the timeout period, the FCCU moves from the Alarm state to the Fault state. The alarm timeout value should be programmed to be less than FOSU_COUNT, or destructive resets may be generated by FOSU timeouts.

NOTE

> This register is writable only when FCCU is in Configuration state.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TO field descriptions

Field	Description
0–31 TO	Noncritical Fault Timeout Timeout = (TO) × T $_{RC16MHz}$

69.6.11 Configuration-State Timeout (FCCU_CFG_TO)

The FCCU_CFG_TO register defines the preset value of the watchdog timer for the recovery from the CONFIG state. Once FCCU enters in CONFIG state, following a SW request (OP1 opcode), the watchdog timer is initialized and starts the countdown if the reset is not asserted.

If the configuration is not completed within the timeout, the FCCU moves automatically from the CONFIG state to the NORMAL state and the default values for all the configuration register is restored. Configuration registers are those registers which can be written only in CONFIG state. The description for a configuration register will contain a NOTE that it is writable only in the CONFIG state. The watchdog timeout is clocked with CLKSAFE. The default timeout value is 4.096 ms. Longer activation of CONFIG state can lead to resets if a failure is indicated during the time the FCCU is in CONFIG state due to FOSU.

NOTE

> The FCCU_CFG_TO register is writable in any state excluding the CONFIG state as follows the execution of the OP1 opcode (NORMAL to CONFIG state) and until the completion of the OP2 opcode (CONFIG to NORMAL state).

In case of watchdog timeout the FCCU_CFG_TO register is not accessible until the OP14 operation (CONFIG to NORMAL) has been completed.

NOTE

> When the configuration watchdog timer expires, FCCU changes the value of this register to its reset value.

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													TO		
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_CFG_TO field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TO	Configuration timeout Timeout = $T_{RC16MHz} \times 2^{(TO + 10)}$ 000 Timeout = 64 µs ... 111 Timeout = 8.192 ms

69.6.12 IO Control (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the EOUT[1] output level when the FCCU is configured in "Test1" or "Test0" fault output mode (FCCU_CFG.FOM)

- to control the EOUT[0] output level when the FCCU is configured in "Test1" or "Test2" fault output mode (FCCU_CFG.FOM)
- to observe the EOUT[1:0] signals in input mode

Table 69-11 shows Bi-stable encoding.

Table 69-11. Bi-stable encoding

Mode = FCCU_CFG.FOM	EOUT[0]	EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

NOTE

Due to the resynchronization stage of the EOUT interface, there is a latency of a few CLKSAFE cycles following a write/read operation of the FCCU_EINOUT register.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										EIN1	EIN0	0		EOUT1	EOUT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_EINOUT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 EIN1	Error input 1. The EIN1 captures the respective fccu_ein[1] input signal. While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[1] pin. 0 when fccu_ein[1] = 0 1 when fccu_ein[1] = 1
27 EIN0	Error input 0.The EIN0 captures the fccu_ein[0] input signal. While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[0] pin. 0 when fccu_ein[0] = 0 1 when fccu_ein[0] = 1
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => EOUT[1] configured in output mode). The EOUT1 set/clear the respective EOUT[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a "don't-care" value. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[1] = 0 1 force EOUT[1] = 1
31 EOUT0	Error out 0 (significative only if the FCCU_CFG.FOM = Test1 or Test2 => EOUT[0] configured in output mode). The EOUT0 set/clear the respective EOUT[0] output signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a "don't care" value. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[0] = 0 1 force EOUT[0] = 1

69.6.13 Status (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine operates by the RC oscillator clock asynchronous with the system clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										PhysicErrorPin		ESTAT	STATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

FCCU_STAT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PhysicErrorPin	PhysicErrorPin : Physical Error Pin's state. PhysicErrorPin[0] corresponds to error pin 0 and PhysicErrorPin[1] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0: Error Pin is at logical 0 1: Error Pin is at logical 1 During Fault state, a static or toggling value is observed based on selected protocol. 0 Error Pin is at logical 0 1 Error Pin is at logical 1

Table continues on the next page...

FCCU_STAT field descriptions (continued)

Field	Description
28 ESTAT	Current Error Pin Status 0 The system is operational (OK). 1 The system is in Fault state. This state is taken from the FCCU EOUT logic (that is, as late as possible).
29–31 STATUS	FCCU Status 000 NORMAL state 001 CONFIG state 010 ALARM state 011 FAULT state 100 Reserved 101 Reserved 110 Reserved 111 Reserved

69.6.14 NA Freeze Status (FCCU_N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the "non-critical fault" source (fccu_ncf[x]) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								NAFS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_N2AF_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 NAFS	Normal to Alarm Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00000000 No transition from NORMAL to ALARM state 00000001 NORMAL to ALARM state transition cause => fccu_ncf[0] fault 00000010 NORMAL to ALARM state transition cause => fccu_ncf[1] fault 00000011 NORMAL to ALARM state transition cause => fccu_ncf[2] fault ... 10000000 NORMAL to ALARM state transition cause => fccu_ncf[127]/ fault 11111111 NORMAL to ALARM state transition cause => multiple fccu_ncf[]/ faults

69.6.15 AF Freeze Status (FCCU_A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (non-critical fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						AF_ SRC		AFFS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_A2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 AF_SRC	Fault source **NOTE:** This field can be read and written but writing 1 has no effect. 00 No fault 01 Reserved 10 "Non-critical" fault 11 Multiple and/or "non-critical" faults
24–31 AFFS	Alarm to Fault Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from ALARM to FAULT state 01h: ALARM to FAULT state transition cause => fccu_ncf[0]/ fault timeout fault 02h: ALARM to FAULT state transition cause => fccu_ncf[1]/ fault timeout fault 03h: ALARM to FAULT state transition cause => fccu_ncf[2]/ fault timeout fault ... 80h: ALARM to FAULT state transition cause => fccu_ncf[127]/ fault timeout fault FFh: ALARM to FAULT state transition cause => multiple fccu_ncf[]/ faults timeout faults

69.6.16 NF Freeze Status (FCCU_N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the source of the non-critical fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						NF_ SRC		NFFS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_N2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 NF_SRC	NF_SRC: Fault source **NOTE:** This field can be read and written but writing 1 has no effect. 00 No fault 01 Reserved 10 "Non-critical" fault 11 Multiple "non-critical" faults
24–31 NFFS	Normal to Fault Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from NORMAL to FAULT state 01h: NORMAL to FAULT state transition cause => fccu_ncf[0] fault 02h: NORMAL to FAULT state transition cause => fccu_ncf[1] fault 03h: NORMAL to FAULT state transition cause => fccu_ncf[2] fault ... 80h: NORMAL to FAULT state transition cause => fccu_ncf[127] fault FFh: NORMAL to FAULT state transition cause => multiple fccu_ncf[] faults

69.6.17 FA Freeze Status (FCCU_F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the source of the non-critical fault (fccu_ncf[x]) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the F2AF_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the F2AF_STATUS register.

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							FAFS								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_F2A_STATUS field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FAFS	Fault to Normal Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from FAULT to ALARM state 01h: FAULT to ALARM state transition cause => fccu_ncf[0]/ fault 02h: FAULT to ALARM state transition cause => fccu_ncf[1]/ fault 03h: FAULT to ALARM state transition cause => fccu_ncf[2]/ fault ... 80h: FAULT to ALARM state transition cause => fccu_ncf[127]/ fault FFh: FAULT to ALARM state transition cause => multiple fccu_ncf[] faults

69.6.18 Noncritical Fault Fake (FCCU_NCFF)

The FCCU_NCFF register contains a unique code to set a non-critical fault in mutually exclusive mode by the external FAULT interface (signal setting). It allows the SW emulation of the non-critical faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The reaction following a fake non-critical fault cannot be masked. The FCCU_NCFF is a write-only register with a set of codes corresponding to each non-critical fault injection.

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																										FNCFC						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFF field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 FNCFC	FNCFC/ : Fake non-critical fault code **NOTE:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results. 00h: Fake non-criticalfault injection at non-criticalfault source 0 01h: Fake non-criticalfault injection at non-criticalfault source 1 02h: Fake non-criticalfault injection at non-criticalfault source 2 ... 7Fh: Fake non-criticalfault injection at non-criticalfault source 127

69.6.19 IRQ Status (FCCU_IRQ_STAT)

The FCCU_IRQ_STAT register provides the FCCU interrupt status related to the following events:

- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set. It is cleared when a 1 is written to the CFG_TO_STAT bit.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											Reserved	Reserved	NMI_STAT	ALRM_STAT	CFG_TO_STAT
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_STAT field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved.
28 Reserved	This field is reserved.
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

69.6.20 IRQ Enable (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												0	0		CFG_TO_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_EN field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled 1 Configuration timeout interrupt enabled

69.6.21 XTMR (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer. These timers are clocked by CLKSAFE.

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
    - OP17
    - OP19
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_XTMR register.

Table 69-12 shows Timer state/value.

Table 69-12. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_XTMR field descriptions

Field	Description
0–31 XTMR	Alarm/Watchdog/Safe request timer The current timer value is measured in CLKSAFE cycles. These bits can be read by the software.

69.6.22 Mode Controller Status (FCCU_MCS)

Indicates the last four chip modes—as defined by the MC_ME module—and whether FCCU was in Fault state when FCCU captured each mode.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0		MCS3				VL2	FS2	0		MCS2			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0		MCS1				VL0	FS0	0		MCS0			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.

FCCU_MCS field descriptions

Field	Description
0 VL3	Valid 3—Indicates whether MCS3 and FS3 are valid. 0 Invalid 1 Valid
1 FS3	Fault Status 3—Indicates whether FCCU was in Fault state when FCCU captured MCS3. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 A state other than Fault state 1 Fault state
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MCS3	Mode Controller State 3—Indicates the fourth most recent chip mode.
8 VL2	Valid 2—Indicates whether MCS2 and FS2 are valid. 0 Invalid 1 Valid
9 FS2	Fault Status 2—Indicates whether FCCU was in Fault state when FCCU captured MCS2. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 A state other than Fault state 1 Fault state
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 MCS2	Mode Controller State 2—Indicates the third most recent chip mode.
16 VL1	Valid 1—Indicates whether MCS1 and FS1 are valid. 0 Invalid 1 Valid
17 FS1	Fault Status 1—Indicates whether FCCU was in Fault state when FCCU captured MCS1. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 A state other than Fault state 1 Fault state
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 MCS1	Mode Controller State 1—Indicates the second most recent chip mode.
24 VL0	Valid 0—Indicates whether MCS0 and FS0 are valid.

Table continues on the next page...

FCCU_MCS field descriptions (continued)

Field	Description
	0 Invalid 1 Valid
25 FS0	Fault Status 0—Indicates whether FCCU was in Fault state when FCCU captured MCS0. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 A state other than Fault state 1 Fault state
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MCS0	Mode Controller State 0—Indicates the most recent (current) chip mode.

69.6.23 Transient Lock (FCCU_TRANS_LOCK)

The FCCU_TRANS register is used for unlocking configuration by writing BCh. This register is available only in supervisor mode. This register is write only.

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																								TRANSKEY								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_TRANS_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 TRANSKEY	Transition Unlocking Key value BCh: Configuration Unlocked Any other value:FCCU gets transiently locked again.

69.6.24 Permanent Lock (FCCU_PERMNT_LOCK)

The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing FFh. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																								PERMNTKEY								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_PERMNT_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 PERMNTKEY	Transition Locking Key value FFh: Configuration Locked Any other value: No Change

69.6.25 Delta T (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds.

NOTE

This register can be written only when the FCCU is in CONFIG state.

NOTE

Reserved bits should always be written as all 0's.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved		Reserved													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		DELTA_T													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_DELTA_T field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–15 Reserved	This field is reserved.
16–17 Reserved	This field is reserved.
18–31 DELTA_T	Controls the minimum amount of time (T_min) that the EOUT fault-indication timer runs according to this equation: T_min = 250 µs + DELTA_T FCCU starts the EOUT fault-indication timer when all of the following are true: • The EOUT fault-indication timer isn't already running. • FCCU enters Fault state as the result of a fault. When the EOUT fault-indication timer is already running and a new fault occurs: • When FCCU is in Configuration state: FCCU doesn't restart the EOUT fault-indication timer. • When FCCU is in Normal state or Alarm state: • And Alarm state is enabled for the fault (noncritical): FCCU enters (or remains in) Alarm state but doesn't restart the EOUT fault-indication timer. • And Alarm state is disabled for the fault (noncritical) or not applicable: FCCU enters Fault state and restarts the EOUT fault-indication timer. • When FCCU is in Fault state: FCCU restarts the EOUT fault-indication timer. FCCU stops and reinitializes the EOUT fault-indication timer when all of the following are true: • T_min has expired. • All faults that caused FCCU to enter or remain in Fault state since FCCU started the EOUT fault-indication timer have been cleared, causing FCCU to return to Normal state. The following values and meanings are for DELTA_T: 00_0000_0000_0000: 0 µs 00_0000_0000_0001: 1 µs … 11_1111_1111_1110: 16,382 µs 11_1111_1111_1111: 16,383 µs **NOTE:** The specified values for DELTA_T may vary ±2% due to the post-trimming inaccuracy of the CLKSAFE clocks.

69.6.26 IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm.

Table 69-13 shows FCCU IRQ alarm enable register channels.

Table 69-13. FCCU IRQ alarm enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0FCh	FCCU_IRQ_ALARM_EN0	IRQENE[31:0]
100h	FCCU_IRQ_ALARM_EN1	IRQENE[63:32]
104h	FCCU_IRQ_ALARM_EN2	IRQENE[95:64]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + FCh offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IRQENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0–31 IRQENx	IRQ alarm enable See Table 69-13 for register offset to channel number relationship. 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

69.6.27 NMI Enable (FCCU_NMI_EN*n*)

These registers enable the NMI.

NOTE

Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.

Table 69-14 shows FCCU NMI enable register channels.

Table 69-14. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
10Ch	FCCU_NMI_EN0	NMIENE[31:0]
110h	FCCU_NMI_EN1	NMIENE[63:32]
114h	FCCU_NMI_EN2	NMIENE[95:64]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + 10Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NMIENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NMI_EN*n* field descriptions

Field	Description
0–31 NMIENx	NMI enable See Table 69-14 for register offset to channel number relationship. 0 NMI is disabled for error (NCF) source x. 1 NMI is enabled for error (NCF) source x.

69.6.28 Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN*n*)

Writable only when FCCU is in Configuration state. Controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channels when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

		EOUTENx fields	
Register	Offset	Most significant (leftmost)	Least significant (rightmost)
FCCU_EOUT_SIG_EN0	11Ch	EOUTEN31	EOUTEN0
FCCU_EOUT_SIG_EN1	120h	EOUTEN63	EOUTEN32
FCCU_EOUT_SIG_EN2	124h	EOUTEN95	EOUTEN64

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EOUTENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0–31 EOUTENx	Fault-Output (EOUT) Signaling Enable Writable only when FCCU is in Configuration state. Controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x) when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]). Fault-output signaling is always enabled for the associated channel, regardless of the value of this field, when FCCU is configured for any other fault-output mode. **NOTE:** To ensure that the FOSU module is aware that fault-output signaling is enabled as a reaction for the associated noncritical fault channel (so that FOSU doesn’t mistakenly generate a destructive chip reset), you must set this field to enabled when FCCU is configured for any mode other than Bistable fault-output mode. For the mapping of the EOUTENx fields among registers, see the earlier table in this register description. 0 Disabled 1 Enabled

69.7 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary fault notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after a fault is signaled.

The intention of the FOSU is to provide a secondary fault reaction path in most cases when the FCCU fails but not to needlessly propagate a fault which is already handled by the FCCU in a full chip reset. Only a failed primary fault reaction (that is, FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction for ALL FAULTS. It is a "static" input in the sense that it does not change after FCCU configuration. The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted, (i.e., a disabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state. Also, the case of a continuously incoming disabled fault being enabled later, is not monitored.

The FOSU contains a timer with a duration of FOSU_COUNT, driven by CLKSAFE. The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FOSU does not check whether the reaction is the configured one for the faults which occurred):

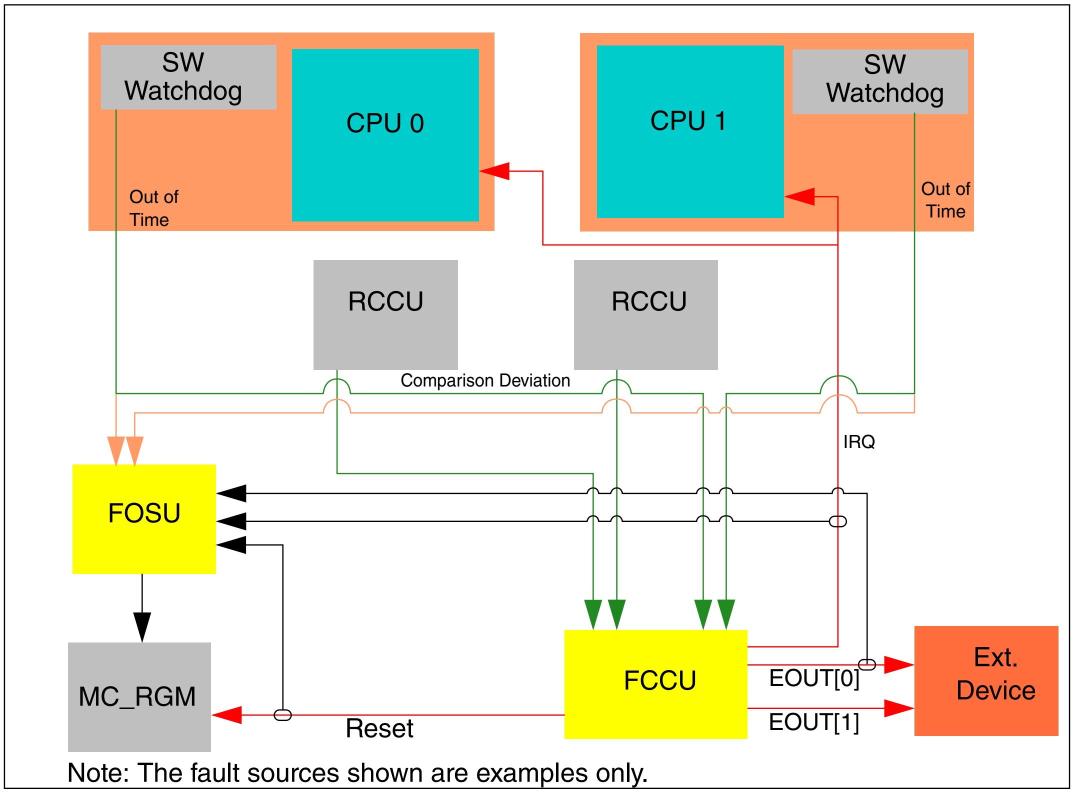
- Reset: Long or short functional reset
- IRQ: NMI or Alarm
- Error out triggered (by FCCU or by SW)

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still 'stuck' in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static "fccu program to do nothing" signal is deasserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.

The FOSU's failure indicator output is connected to one of the MC_RGM's 'destructive' reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as is used by the FCCU, which is asserted on power-on, 'destructive', and external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

NOTE

FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction. All intermediate faults are masked (not monitored) during this timeout cycle.



Note: The fault sources shown are examples only.

Figure 69-14. FOSU connections to the FCCU and MC_RGM

It is important to remember that there will be no FCCU reaction to a fault while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

NOTE

FCCU and FOSU timeout counter settings: The FCCU counter value should always be programmed less than the FOSU counter value, FOSU_COUNT, or destructive resets may be generated by an FOSU time-out. This ensures that the FOSU reacts only when the FCCU fails to react to any particular fault.

69.8 Use cases and limitations

69.8.1 Misconfigurations

The following configurations are appropriate:

1. If at least one reaction to fault is enabled by default then the corresponding fault should also be enabled by default.
2. Enabling a fault but disabling all reactions is not a meaningful configuration from safety point of view.
3. Disable the fault and also its reactions. Applicable when a specific fault line is not of interest in a specific application scenario

   Example: FCCU goes into ALARM state due to a failure, but the IRQ is disabled for that failure.
4. If the user programs the FCCU for a higher timeout value than what is hardcoded for FOSU, then FCCU reacts later than what FOSU would be expecting and, as a result, FOSU will generate destructive reset request on its timeout.

69.8.2 Recommendations to configure FCCU

1. After a power on, 'destructive', or long external reset, where both system and FCCU are reset, the following steps could be followed to configure FCCU:
   a. Check and clear any pending fault status
   b. Verify FCCU is in NORMAL state, else repeat step(a) above
   c. Configure FCCU
2. After a short external or any 'functional' reset of the system, arising out of a reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a. If active, wait for the Error out T_min to expire
   b. Check and clear fault status
   c. Error pin moves to "non faulty" state, once fault status is cleared and T_min expires
   d. Verify FCCU is in NORMAL state, else repeat step(a) above
   e. Read and verify value in FCCU_NCF_Ex
   f. Reconfigure FCCU, if necessary

Chapter 70
Self-Test Control Unit (STCU2)

70.1 Introduction

The Self-Test Control Unit (STCU2) controls the execution of the built-in self-test of the MPC5744P.

The built-in self-test supports in the application:

- Memory (RAM/ROM) Built-In Self-Test (MBIST) at startup and shutdown.
- Scan-based Logic BIST for digital logic (LBIST), which is divided into multiple partitions, at startup and shutdown.

The following table summarizes the three self-test modes. For detailed descriptions of the modes, see Use cases and limitations.

Table 70-1. STCU2 self-test modes

Self-test mode	Description	Availability
Normal self-test mode	MBIST for all memories with a checkerboard/inverted checkerboard algorithm LBIST on all partitions	In startup mode and shutdown mode
Long self-test mode	MBIST for all memories with the reduced RAM and ROM algorithm LBIST on all partitions	Only during shutdown mode
Diagnostic mode	MBIST for all memories with the full RAM and ROM algorithm LBIST on all partitions	Only during shutdown mode

In application mode, the STCU2 initiates self-test execution:

- after Power-on Reset (POR)
- after external reset (if triggered as long-functional reset)
- after destructive reset
- when initiated by software

The STCU2 communicates any fault information to the Fault Control and Collection Unit (FCCU) and to the Reset Generation Module (MC_RGM).

The startup self-test is implemented by Freescale. The user can choose either to run normal self-test execution during startup of the device or to bypass the startup self-test execution by programming the appropriate records in the UTEST flash memory.

The shutdown self-test sequence is user programmable and must be started by the user application on demand. This self-test can be executed at less critical times in the application. After the self-test is executed during shutdown, a long functional reset is initiated, and the self-test results are accessible.

The STCU2 includes the SSCM DCF bus to load the self-test parameters from flash memory—consisting of both the test flash memory, which is programmed by Freescale, and the UTEST flash, which is user-programmable—during the startup self-test phase.

70.2 Functional description

70.2.1 FCCU interface

The FCCU interface is the hardware flag mechanism towards the system that indicates detection of an STCU2 critical fault and/or an STCU2 non-critical fault during the self-test sequence.

The FCCU interface allows the user application to check the integrity of the critical fault and non-critical fault connection lines between the STCU2 and the FCCU. See the description of FCCU fault injection mechanism to understand how this set/clear mechanism works.

70.2.2 Watchdogs

The STCU2 implements three different watchdog ensure that operations are finished in time.

70.2.2.1 Initialization sequence

The STCU2 has two initialization operating modes: program the self-test sequence and run it or bypass completely the self-test sequence. In case there are faults during the initialization phase preventing the selection of one of these two operating modes, a hard-coded Watchdog time-out flags the wrong behavior.

70.2.2.2 LBIST and MBIST scheduling

If the selected LBISTs or MBISTs are not completed during the self-test's assigned time, the current execution of LBISTs or MBISTs is interrupted and a failure is flagged in:

- STCU_ERR_STAT[WDTO] and STCU_MBEL or STCU_LBE for a startup self-test
- STCU_ERR_STAT[WDTOSW] and STCU_MBELSW or STCU_LBESW for a shutdown self-test

NOTE

The only sequence possible is to run MBISTs first and then LBISTs.

70.2.2.3 Write access time-out

The access on the STCU2 registers during the self-test configuration phase (both shutdown and startup) is protected by a security key mechanism to prevent any unwanted write access, as described in the STCU_SKC register description. In addition to this safety feature, there is a hardware Watchdog that, after 4096 STCU2 clock cycles, locks access. In that event, the user must re-apply the STCU_SKC keys.

70.2.3 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters during the startup sequence without CPU intervention after a reset trigger event initializes the STCU2.

70.3 Register description: configuration registers

Use these registers to configure self-tests.

The reset value of each of these registers can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

These registers of the STCU2 module are accessible (read/write) in each access mode: user or supervisor.

The following read operations (contiguous byte enables) are supported:

- Word (32-bit) data read operations to any registers

- Low and high half-word (16-bit, data[31:16] or data[15:0]) data read operations to any registers
- Byte (8-bit, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/half-word/word (free byte enables or other operations) on each register
- Any write operation on Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Security keys have not been applied
- Any write operation performed on read-only registers

Write operations to fields marked as reserved do not generate the transfer error.

STCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	STCU2 Run Software Register (STCU_RUNSW)	32	R/W	See section	70.3.1/2782
8	STCU2 SK Code Register (STCU_SKC)	32	W	See section	70.3.2/2782
C	STCU2 Configuration Register (STCU_CFG)	32	R/W	See section	70.3.3/2783
14	STCU2 Watchdog Register Granularity (STCU_WDG)	32	R/W	See section	70.3.4/2783
3C	STCU2 Shutdown LBIST Reset Management (STCU_LBRMSW)	32	R/W	See section	70.3.5/2784
100	STCU2 LBIST Control Register (STCU_LB0_CTRL)	32	R/W	See section	70.3.6/2784
104	STCU2 LBIST PC Stop Register (STCU_LB0_PCS)	32	R/W	See section	70.3.7/2784
120	STCU2 Shutdown LBIST MISR Expected Low Register (STCU_LB0_MISRELSW)	32	R/W	See section	70.3.8/2784
124	STCU2 Shutdown LBIST MISR Expected High Register (STCU_LB0_MISREHSW)	32	R/W	See section	70.3.9/2785
140	STCU2 LBIST Control Register (STCU_LB1_CTRL)	32	R/W	See section	70.3.6/2784
144	STCU2 LBIST PC Stop Register (STCU_LB1_PCS)	32	R/W	See section	70.3.7/2784
160	STCU2 Shutdown LBIST MISR Expected Low Register (STCU_LB1_MISRELSW)	32	R/W	See section	70.3.8/2784
164	STCU2 Shutdown LBIST MISR Expected High Register (STCU_LB1_MISREHSW)	32	R/W	See section	70.3.9/2785
180	STCU2 LBIST Control Register (STCU_LB2_CTRL)	32	R/W	See section	70.3.6/2784
184	STCU2 LBIST PC Stop Register (STCU_LB2_PCS)	32	R/W	See section	70.3.7/2784
1A0	STCU2 Shutdown LBIST MISR Expected Low Register (STCU_LB2_MISRELSW)	32	R/W	See section	70.3.8/2784

Table continues on the next page...

STCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A4	STCU2 Shutdown LBIST MISR Expected High Register (STCU_LB2_MISREHSW)	32	R/W	See section	70.3.9/2785
1C0	STCU2 LBIST Control Register (STCU_LB3_CTRL)	32	R/W	See section	70.3.6/2784
1C4	STCU2 LBIST PC Stop Register (STCU_LB3_PCS)	32	R/W	See section	70.3.7/2784
1E0	STCU2 Shutdown LBIST MISR Expected Low Register (STCU_LB3_MISRELSW)	32	R/W	See section	70.3.8/2784
1E4	STCU2 Shutdown LBIST MISR Expected High Register (STCU_LB3_MISREHSW)	32	R/W	See section	70.3.9/2785
600	STCU2 MBIST Control Register (STCU_MB0_CTRL)	32	R/W	See section	70.3.10/ 2785
604	STCU2 MBIST Control Register (STCU_MB1_CTRL)	32	R/W	See section	70.3.10/ 2785
608	STCU2 MBIST Control Register (STCU_MB2_CTRL)	32	R/W	See section	70.3.10/ 2785
60C	STCU2 MBIST Control Register (STCU_MB3_CTRL)	32	R/W	See section	70.3.10/ 2785
610	STCU2 MBIST Control Register (STCU_MB4_CTRL)	32	R/W	See section	70.3.10/ 2785
614	STCU2 MBIST Control Register (STCU_MB5_CTRL)	32	R/W	See section	70.3.10/ 2785
618	STCU2 MBIST Control Register (STCU_MB6_CTRL)	32	R/W	See section	70.3.10/ 2785
61C	STCU2 MBIST Control Register (STCU_MB7_CTRL)	32	R/W	See section	70.3.10/ 2785
620	STCU2 MBIST Control Register (STCU_MB8_CTRL)	32	R/W	See section	70.3.10/ 2785
624	STCU2 MBIST Control Register (STCU_MB9_CTRL)	32	R/W	See section	70.3.10/ 2785
628	STCU2 MBIST Control Register (STCU_MB10_CTRL)	32	R/W	See section	70.3.10/ 2785
62C	STCU2 MBIST Control Register (STCU_MB11_CTRL)	32	R/W	See section	70.3.10/ 2785
630	STCU2 MBIST Control Register (STCU_MB12_CTRL)	32	R/W	See section	70.3.10/ 2785
634	STCU2 MBIST Control Register (STCU_MB13_CTRL)	32	R/W	See section	70.3.10/ 2785
638	STCU2 MBIST Control Register (STCU_MB14_CTRL)	32	R/W	See section	70.3.10/ 2785
63C	STCU2 MBIST Control Register (STCU_MB15_CTRL)	32	R/W	See section	70.3.10/ 2785
640	STCU2 MBIST Control Register (STCU_MB16_CTRL)	32	R/W	See section	70.3.10/ 2785
644	STCU2 MBIST Control Register (STCU_MB17_CTRL)	32	R/W	See section	70.3.10/ 2785

Table continues on the next page...

STCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
648	STCU2 MBIST Control Register (STCU_MB18_CTRL)	32	R/W	See section	70.3.10/ 2785
64C	STCU2 MBIST Control Register (STCU_MB19_CTRL)	32	R/W	See section	70.3.10/ 2785
650	STCU2 MBIST Control Register (STCU_MB20_CTRL)	32	R/W	See section	70.3.10/ 2785
654	STCU2 MBIST Control Register (STCU_MB21_CTRL)	32	R/W	See section	70.3.10/ 2785
658	STCU2 MBIST Control Register (STCU_MB22_CTRL)	32	R/W	See section	70.3.10/ 2785
65C	STCU2 MBIST Control Register (STCU_MB23_CTRL)	32	R/W	See section	70.3.10/ 2785
660	STCU2 MBIST Control Register (STCU_MB24_CTRL)	32	R/W	See section	70.3.10/ 2785
664	STCU2 MBIST Control Register (STCU_MB25_CTRL)	32	R/W	See section	70.3.10/ 2785
668	STCU2 MBIST Control Register (STCU_MB26_CTRL)	32	R/W	See section	70.3.10/ 2785

70.3.1 STCU2 Run Software Register (STCU_RUNSW)

This register initiates the shutdown self-test.

The reset value of this register can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to this register for each shutdown self-test.

70.3.2 STCU2 SK Code Register (STCU_SKC)

The STCU_SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers. In order to unlock the STCU2 for shutdown self-test the software (IPS bus) must apply the following sequence:

1. write key1 into the STCU_SKC register
2. write key2 into the STCU_SKC register

A byte write operation is not allowed because the full key must be recognized as one unit.

See Table 70-4 for the key values to write to this register for each shutdown self-test.

The reset value of this register can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

The STCU_SKC register is not readable. A read operation always returns the value 00000000h.

NOTE

After the self-test sequence has been completed, the SSCM interface is no longer available.

In case of invalid access or sequence (key1 and key2 must be applied consecutively), a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked. To unlock the access, re-apply the sequence.

If the STCU2 register access lasts for longer than 4096 cycles, the STCU2 write access is locked as described in Write access time-out. In this case, to re-enable the write access to the STCU2, apply the sequence again.

In case it is required to extend the STCU2 register access cycles before the hard-coded WDG time-out expires, apply only key2. The effect of this write operation is to re-initialize the WDG time-out counter. In such a condition, do not apply key1. If it is applied, a transfer error on the IPS or SSCM bus is asserted depending on the selected source, and the STCU2 write access is locked. To unlock the access, re-apply the sequence.

70.3.3 STCU2 Configuration Register (STCU_CFG)

This register includes global configuration controls for the STCU2.

The reset value of this register can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to this register for each shutdown self-test.

70.3.4 STCU2 Watchdog Register Granularity (STCU_WDG)

This register defines the time budget of LBIST and MBIST execution.

The reset value of this register can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to this register for each shutdown self-test.

70.3.5 STCU2 Shutdown LBIST Reset Management (STCU_LBRMSW)

The STCU_LBRMSW register enables and ensures the correct reset sequence after shutdown self-test execution.

The reset value of this register can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to this register for each shutdown self-test.

70.3.6 STCU2 LBIST Control Register (STCU_LB*n*_CTRL)

The STCU_LBn_CTRL register defines the control settings of the corresponding LBIST.

The reset value of each of these registers can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to each register for each shutdown self-test.

70.3.7 STCU2 LBIST PC Stop Register (STCU_LB*n*_PCS)

The STCU_LBn_PCS register contains pattern-related information.

The reset value of each of these registers can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to each register for each shutdown self-test.

70.3.8 STCU2 Shutdown LBIST MISR Expected Low Register (STCU_LB*n*_MISRELSW)

The STCU_LBn_MISRELSW register contains MISR signature-related information.

The reset value of each of these registers can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to each register for each shutdown self-test.

70.3.9 STCU2 Shutdown LBIST MISR Expected High Register (STCU_LB*n*_MISREHSW)

The STCU_LBn_MISREHSW register contains MISR signature-related information.

The reset value of each of these registers can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to each register for each shutdown self-test.

70.3.10 STCU2 MBIST Control Register (STCU_MB*n*_CTRL)

The STCU_MBn_CTRL register contains control settings for the corresponding MBIST.

The reset value of each of these registers can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to each register for each shutdown self-test.

70.4 Register description: clock configuration register

DCL memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FFC0_0104	Different Clock register (DCL_IPS0)	32	R/W	See section	70.4.1/2785

70.4.1 Different Clock register (DCL_IPS0)

This register selects different clock configurations for the self-test.

The reset value of this register can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

See Table 70-4 for the value to write to this register for each shutdown self-test.

70.5 Register description: status registers

Read these registers to determine the status and results of executed self-tests.

The reset value of each of these registers can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

These registers of the STCU2 module are accessible in each access mode: user or supervisor.

The following read operations (contiguous byte enables) are supported:

- Word (32-bit) data read operations to any registers
- Low and high half-word (16-bit, data[31:16] or data[15:0]) data read operations to any registers
- Byte (8-bit, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/half-word/word (free byte enables or other operations) on each register
- Any write operation on Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Security keys have not been applied
- Any write operation performed on read-only registers

Write operations to fields marked as reserved do not generate the transfer error.

STCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24	STCU2 Error Register (STCU_ERR_STAT)	32	R/W	See section	70.5.1/2787
2C	STCU2 Startup LBIST Status Register (STCU_LBS)	32	R	See section	70.5.2/2790
30	STCU2 Startup LBIST End Flag Register (STCU_LBE)	32	R	See section	70.5.3/2791
34	STCU2 Shutdown LBIST Status Register (STCU_LBSSW)	32	R	See section	70.5.4/2792
38	STCU2 Shutdown LBIST End Flag Register (STCU_LBESW)	32	R	See section	70.5.5/2793

Table continues on the next page...

STCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
44	STCU2 Startup MBIST Status Low Register (STCU_MBSL)	32	R	See section	70.5.6/2795
50	STCU2 Startup MBIST End Flag Low Register (STCU_MBEL)	32	R	See section	70.5.7/2799
5C	STCU2 Shutdown MBIST Status Low Register (STCU_MBSLSW)	32	R	See section	70.5.8/2803
68	STCU2 Shutdown MBIST End Flag Low Register (STCU_MBELSW)	32	R	See section	70.5.9/2807

70.5.1 STCU2 Error Register (STCU_ERR_STAT)

The STCU_ERR_STAT register includes the status flags related to the STCU2 internal error conditions occurred during the configuration or the self-test execution.

The CFSF and NCFSF can be set/cleared using the FCCU dedicated channels.

The access to this register is described in the figure and as follows:

- If you select the byte write capability to write only the CFSF and NCFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides CFSF/NCFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate the CFSF/NCFSF single bit set/clean capability.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						0	0	0			LOCKESW	WDTOSW	0	ENGESW	INVPSW
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CFSF	NCFSF	0			LOCKE	WDTO	0	ENGE	INVP
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_ERR_STAT field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 LOCKESW	Shutdown LOCK Error You can always read this field. The content of this field is initialized to its reset value as soon as a shutdown self-test is initiated. 0 The PLL was correctly locked during the self-test sequence 1 This flag highlights that an unexpected PLL unlock event occurred during the shutdown self-test sequence execution. The shutdown self-test run was stopped and the status of the currently running LBISTs or MBISTs was saved into the related registers.
12 WDTOSW	Shutdown Watchdog time-out You can always read this field. The content of this field is initialized to its reset value as soon as a shutdown self-test is initiated. 0 LBIST and MBIST time slots have been completed within the assigned watchdog time. 1 LBIST and MBIST time slots have not been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The possible conditions that flag the failures are the following: • At least one MBIST did not finish • At least one LBIST did not finish
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

STCU_ERR_STAT field descriptions (continued)

Field	Description
14 ENGESW	Shutdown Engine Error You can always read this field. The content of this field is initialized to its reset value as soon as a shutdown self-test is initiated. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) select an unused code • Pass/fail status between MBIST logic and STCU is not aligned • MBIST finished status between MBIST logic and STCU is not aligned
15 INVPSW	Shutdown Invalid Pointer You can always read this field. The content of this field is initialized to its reset value as soon as a shutdown self-test is initiated. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 CFSF	Critical Faults Status Flag This flag reports the global status of the CF. 0 No errors that trigger the Critical Faults condition 1 There are errors that trigger the Critical Faults condition
23 NCFSF	Non-Critical Faults Status Flag 0 No errors that trigger the Non-Critical Faults condition 1 There are errors that trigger the Non-Critical Faults condition
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKE	Startup LOCK Error 0 The PLL was correctly locked during the startup self-test sequence 1 This flag highlights that an unexpected PLL unlock event occurred during the startup self-test sequence execution. The startup self-test run was stopped and the status of the currently running LBISTs or MBISTs was saved into the related registers.
28 WDTO	Startup Watchdog time-out 0 LBIST and MBIST time slots have been completed within the assigned watchdog time. 1 LBIST and MBIST time slots have not been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • The startup self-test execution was not prevented correctly. (This condition is possible for only the "No self-test execution" startup self-test selection.) • The STCU2 configuration through the SSCM was interrupted. The startup self-test did not execute properly. (This condition is possible for only the "Normal startup self-test" startup self-test selection.) • At least one MBIST did not finish • At least one LBIST did not finish

Table continues on the next page...

STCU_ERR_STAT field descriptions (continued)

Field	Description
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ENGE	Startup Engine Error 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) select an unused code • Pass/fail status between MBIST logic and STCU is not aligned • MBIST finished status between MBIST logic and STCU is not aligned
31 INVP	Startup Invalid pointer 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)

70.5.2 STCU2 Startup LBIST Status Register (STCU_LBS)

The STCU_LBS register includes the results corresponding to the execution of the startup LBIST.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LBS3	LBS2	LBS1	LBS0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_LBS field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBS3	Startup status of the LBIST3

Table continues on the next page...

STCU_LBS field descriptions (continued)

Field	Description
	0 Failed LBIST execution 1 Successful LBIST execution
29 LBS2	Startup status of the LBIST2 0 Failed LBIST execution 1 Successful LBIST execution
30 LBS1	Startup status of the LBIST1 0 Failed LBIST execution 1 Successful LBIST execution
31 LBS0	Startup status of the LBIST0 0 Failed LBIST execution 1 Successful LBIST execution

70.5.3 STCU2 Startup LBIST End Flag Register (STCU_LBE)

The STCU_LBE register includes the End Flag related to the execution of the startup LBIST.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LBE3	LBE2	LBE1	LBE0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_LBE field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBE3	Startup End status of the LBIST3

Table continues on the next page...

STCU_LBE field descriptions (continued)

Field	Description
	0 LBIST execution not yet completed 1 LBIST execution finished
29 LBE2	Startup End status of the LBIST2 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBE1	Startup End status of the LBIST1 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBE0	Startup End status of the LBIST0 0 LBIST execution not yet completed 1 LBIST execution finished

70.5.4 STCU2 Shutdown LBIST Status Register (STCU_LBSSW)

The STCU_LBSSW register includes the results corresponding to the execution of the shutdown LBIST.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LBSSW3	LBSSW2	LBSSW1	LBSSW0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_LBSSW field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBSSW3	Shutdown status of the LBIST3 0 Failed LBIST execution 1 Successful LBIST execution
29 LBSSW2	Shutdown status of the LBIST2 0 Failed LBIST execution 1 Successful LBIST execution
30 LBSSW1	Shutdown status of the LBIST1 0 Failed LBIST execution 1 Successful LBIST execution
31 LBSSW0	Shutdown status of the LBIST0 0 Failed LBIST execution 1 Successful LBIST execution

70.5.5 STCU2 Shutdown LBIST End Flag Register (STCU_LBESW)

The STCU_LBESW register includes the End Flag related to the execution of the shutdown LBIST.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LBESW3	LBESW2	LBESW1	LBESW0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_LBESW field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBESW3	Shutdown End status of the LBIST3 0 LBIST execution not yet completed 1 LBIST execution finished
29 LBESW2	Shutdown End status of the LBIST2 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBESW1	Shutdown End status of the LBIST1 0 LBIST execution not yet completed 1 LBIST execution finished

Table continues on the next page...

STCU_LBESW field descriptions (continued)

Field	Description
31 LBESW0	Shutdown End status of the LBIST0 0 LBIST execution not yet completed 1 LBIST execution finished

70.5.6 STCU2 Startup MBIST Status Low Register (STCU_MBSL)

The STCU_MBSL register includes the results corresponding to the execution of the startup MBIST.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					MBS26	MBS25	MBS24	MBS23	MBS22	MBS21	MBS20	MBS19	MBS18	MBS17	MBS16
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS15	MBS14	MBS13	MBS12	MBS11	MBS10	MBS9	MBS8	MBS7	MBS6	MBS5	MBS4	MBS3	MBS2	MBS1	MBS0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_MBSL field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MBS26	Startup status of the MBIST26 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
6 MBS25	Startup status of the MBIST25 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
7 MBS24	Startup status of the MBIST24 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
8 MBS23	Startup status of the MBIST23 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
9 MBS22	Startup status of the MBIST22 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
10 MBS21	Startup status of the MBIST21 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
11 MBS20	Startup status of the MBIST20 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
12 MBS19	Startup status of the MBIST19

Table continues on the next page...

STCU_MBSL field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
13 MBS18	Startup status of the MBIST18 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
14 MBS17	Startup status of the MBIST17 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
15 MBS16	Startup status of the MBIST16 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
16 MBS15	Startup status of the MBIST15 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
17 MBS14	Startup status of the MBIST14 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
18 MBS13	Startup status of the MBIST13 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
19 MBS12	Startup status of the MBIST12 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution

Table continues on the next page...

STCU_MBSL field descriptions (continued)

Field	Description
20 MBS11	Startup status of the MBIST11 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
21 MBS10	Startup status of the MBIST10 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
22 MBS9	Startup status of the MBIST9 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
23 MBS8	Startup status of the MBIST8 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
24 MBS7	Startup status of the MBIST7 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
25 MBS6	Startup status of the MBIST6 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
26 MBS5	Startup status of the MBIST5 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
27 MBS4	Startup status of the MBIST4 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished.

Table continues on the next page...

STCU_MBSL field descriptions (continued)

Field	Description
	0 Failed BIST execution 1 No Fault detected during the BIST execution
28 MBS3	Startup status of the MBIST3 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
29 MBS2	Startup status of the MBIST2 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
30 MBS1	Startup status of the MBIST1 **NOTE:** This bit is meaningful when the related bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
31 MBS0	Startup status of the MBIST0 This bit has no significance. The MBIST0 logic involves the BAM ROM. The 4N checkerboard algorithm (as it is processed during normal startup self-tests) is not executed on MBIST0.

70.5.7 STCU2 Startup MBIST End Flag Low Register (STCU_MBEL)

The STCU_MBEL register includes the End Flag related to the execution of the startup MBIST.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_MBEL field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MBE26	Startup End status of the MBIST26 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE25	Startup End status of the MBIST25 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE24	Startup End status of the MBIST24 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU_MBEL field descriptions (continued)

Field	Description
8 MBE23	Startup End status of the MBIST23 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE22	Startup End status of the MBIST22 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE21	Startup End status of the MBIST21 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE20	Startup End status of the MBIST20 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE19	Startup End status of the MBIST19 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE18	Startup End status of the MBIST18 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBE17	Startup End status of the MBIST17 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE16	Startup End status of the MBIST16 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE15	Startup End status of the MBIST15 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE14	Startup End status of the MBIST14 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE13	Startup End status of the MBIST13 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE12	Startup End status of the MBIST12 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU_MBEL field descriptions (continued)

Field	Description
20 MBE11	Startup End status of the MBIST11 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE10	Startup End status of the MBIST10 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE9	Startup End status of the MBIST9 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE8	Startup End status of the MBIST8 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE7	Startup End status of the MBIST7 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE6	Startup End status of the MBIST6 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE5	Startup End status of the MBIST5 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE4	Startup End status of the MBIST4 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE3	Startup End status of the MBIST3 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE2	Startup End status of the MBIST2 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE1	Startup End status of the MBIST1 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE0	Startup End status of the MBIST0 This bit has no significance. The MBIST0 logic involves the BAM ROM. The 4N checkerboard algorithm (as it is processed during normal startup self-tests) is not executed on MBIST0.

70.5.8 STCU2 Shutdown MBIST Status Low Register (STCU_MBSLSW)

The STCU_MBSLSW register includes the results corresponding to the execution of the shutdown MBIST.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					MBSSW26	MBSSW25	MBSSW24	MBSSW23	MBSSW22	MBSSW21	MBSSW20	MBSSW19	MBSSW18	MBSSW17	MBSSW16
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15	MBSSW14	MBSSW13	MBSSW12	MBSSW11	MBSSW10	MBSSW9	MBSSW8	MBSSW7	MBSSW6	MBSSW5	MBSSW4	MBSSW3	MBSSW2	MBSSW1	MBSSW0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_MBSLSW field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MBSSW26	Shutdown status of the MBIST26 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
6 MBSSW25	Shutdown status of the MBIST25 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
7 MBSSW24	Shutdown status of the MBIST24 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
8 MBSSW23	Shutdown status of the MBIST23 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
9 MBSSW22	Shutdown status of the MBIST22 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
10 MBSSW21	Shutdown status of the MBIST21 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
11 MBSSW20	Shutdown status of the MBIST20 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
12 MBSSW19	Shutdown status of the MBIST19

Table continues on the next page...

STCU_MBSLSW field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
13 MBSSW18	Shutdown status of the MBIST18 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
14 MBSSW17	Shutdown status of the MBIST17 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
15 MBSSW16	Shutdown status of the MBIST16 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
16 MBSSW15	Shutdown status of the MBIST15 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
17 MBSSW14	Shutdown status of the MBIST14 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
18 MBSSW13	Shutdown status of the MBIST13 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
19 MBSSW12	Shutdown status of the MBIST12 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution

Table continues on the next page...

STCU_MBSLSW field descriptions (continued)

Field	Description
20 MBSSW11	Shutdown status of the MBIST11 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
21 MBSSW10	Shutdown status of the MBIST10 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
22 MBSSW9	Shutdown status of the MBIST9 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
23 MBSSW8	Shutdown status of the MBIST8 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
24 MBSSW7	Shutdown status of the MBIST7 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
25 MBSSW6	Shutdown status of the MBIST6 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
26 MBSSW5	Shutdown status of the MBIST5 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
27 MBSSW4	Shutdown status of the MBIST4 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished.

Table continues on the next page...

STCU_MBSLSW field descriptions (continued)

Field	Description
	0 Failed BIST execution 1 No Fault detected during the BIST execution
28 MBSSW3	Shutdown status of the MBIST3 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
29 MBSSW2	Shutdown status of the MBIST2 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
30 MBSSW1	Shutdown status of the MBIST1 **NOTE:** This bit is meaningful when the related bit of the STCU_MBELSW register reports the MBIST is finished. 0 Failed BIST execution 1 No Fault detected during the BIST execution
31 MBSSW0	Shutdown status of the MBIST0 **NOTE:** The MBIST0 logic involves the BAM ROM. The 4N checkerboard algorithm (as it is processed during normal shutdown self-tests) does not run on MBIST0. This bit is meaningful: • only after execution of a long self-test mode or of the diagnostic self-test mode • when the related bit of the STCU_MBELSW register reports the MBIST is finished 0 Failed BIST execution 1 No Fault detected during the BIST execution

70.5.9 STCU2 Shutdown MBIST End Flag Low Register (STCU_MBELSW)

The STCU_MBELSW register includes the End Flag related to the execution of the shutdown MBIST.

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					MBESW26	MBESW25	MBESW24	MBESW23	MBESW22	MBESW21	MBESW20	MBESW19	MBESW18	MBESW17	MBESW16
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW15	MBESW14	MBESW13	MBESW12	MBESW11	MBESW10	MBESW9	MBESW8	MBESW7	MBESW6	MBESW5	MBESW4	MBESW3	MBESW2	MBESW1	MBESW0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The register's reset value can vary by device version and due to various dependencies, such as flash memory configuration or the result (pass/fail) of MBISTs and LBISTs.

STCU_MBELSW field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MBESW26	Shutdown End status of the MBIST26 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW25	Shutdown End status of the MBIST25 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU_MBELSW field descriptions (continued)

Field	Description
7 MBESW24	Shutdown End status of the MBIST24 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBESW23	Shutdown End status of the MBIST23 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW22	Shutdown End status of the MBIST22 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW21	Shutdown End status of the MBIST21 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW20	Shutdown End status of the MBIST20 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW19	Shutdown End status of the MBIST19 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW18	Shutdown End status of the MBIST18 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW17	Shutdown End status of the MBIST17 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW16	Shutdown End status of the MBIST16 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW15	Shutdown End status of the MBIST15 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW14	Shutdown End status of the MBIST14 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW13	Shutdown End status of the MBIST13 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU_MBELSW field descriptions (continued)

Field	Description
19 MBESW12	Shutdown End status of the MBIST12 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBESW11	Shutdown End status of the MBIST11 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW10	Shutdown End status of the MBIST10 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW9	Shutdown End status of the MBIST9 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW8	Shutdown End status of the MBIST8 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW7	Shutdown End status of the MBIST7 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW6	Shutdown End status of the MBIST6 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW5	Shutdown End status of the MBIST5 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW4	Shutdown End status of the MBIST4 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW3	Shutdown End status of the MBIST3 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW2	Shutdown End status of the MBIST2 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW1	Shutdown End status of the MBIST1 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU_MBELSW field descriptions (continued)

Field	Description
31 MBESW0	Shutdown End status of the MBIST0 **NOTE:** The MBIST0 logic involves the BAM ROM. The 4N checkerboard algorithm (as it is processed during normal shutdown self-tests) does not run on MBIST0. This bit is meaningful only after execution of a long self-test mode or of the diagnostic self-test mode. 0 MBIST execution still ongoing 1 MBIST execution finished

70.6 Use cases and limitations

The STCU2 module is designed to give a high degree of flexibility by providing multiple self-test types. The following tables summarize the provided STCU2 self-test operations.

NOTE

Make sure that the flash is in IDLE state before starting LBIST (Self Test) operation.

Table 70-2. Startup self-test

Self-test	Coverage	BIST details	Execution time
No self-test execution	No coverage	No LBIST No MBIST	0 ns
Normal startup self-test	Coverage of ≥ 90% stuck-at	LBIST on all partitions MBIST for all memories with a 4N checkerboard algorithm	Max. 18 ms total execution time

Table 70-3. Shutdown self-tests

Self-test	Stuck-at coverage	Transition coverage	LBIST details	MBIST details	Execution time
Normal shutdown self-test	≥ 90% on all partitions	≥ 60% on partition C0 ≥ 60% on partition C1 No transition coverage on partitions P0 and P1	LBIST on all partitions	MBIST for all memories with a 4N checkerboard algorithm	Max. 51 ms total execution time
Long shutdown self-test				MBIST on all memories with reduced RAM and ROM algorithm (without ASOF)	Max. 68 ms total execution time
Diagnostic shutdown self-test	≥ 90% on partition C0 ≥ 90% on partitions C1, P0, and P1	≥ 60% on partition C0 ≥ 60% on partition C1 No transition coverage on partitions P0 and P1		MBIST on all memories with full RAM and ROM algorithm	

70.6.1 Startup self-test sequence

If the normal startup self-test mode is chosen, the system startup sequence includes the following activities if any of the STCU reset conditions is valid.

1. The STCU is configured via DCF during reset phase 3.
2. After the exit from reset phase 3, the STCU powers a self-test setup phase.
3. Self-test executes / MBIST phase
4. Self-test executes / LBIST phase
5. The STCU collects all BIST results and stores them internally.
6. The STCU arranges a long functional reset.

70.6.2 Startup self-test mode selection

This chip offers two startup self-test variants.

- No self-test execution
- Normal startup self-test

For correct system behavior one of these modes must be configured. To configure one of these modes, program particular DCF records in the UTEST flash memory.

To configure "No self-test execution": Store the following DCF record as the first record in the UTEST flash memory.

```
00000400h
00080000h
```

To configure "Normal startup self-test": Store the following DCF record as the first record in the UTEST flash memory.

```
00000301h
00080000h
```

NOTE

During startup self-tests, once the STCU2 registers are configured they cannot be overridden.

70.6.3 Shutdown self-test sequence

This is the typical mode of using the STCU2 module during the application run.

70.6.3.1 Preparation before initiating an STCU shutdown self-test

Before you initiate the execution of an STCU shutdown self-test, you must complete the following prerequisite procedure to prepare the chip.

1. End the currently running application, including graceful termination of peripherals as required by the application.
2. Configure FlexRay and CAN clock sources and divider for self-test:
    a. Configure FRAY_PLL_CLK clock as PLL0 PHI divided by 2: MC_CGM_AC1_DC0[DIV]=1h.
    b. Select FRAY_PLL_CLK as the FRAY_CLK clock source: FR_MCR[CLKSEL]=1.
    c. Select CAN_PLL_CLK as the CAN_CLK clock source: CAN_CTRL1[CLKSRC]=1.
3. Ensure PBRIDGE clock is not over clocked (maximum frequency is 50MHz) when SYS_CLK is switched to 200MHz.
    - Configure PBRIDGE*_CLK as SYS_CLK (system clock) divided by 4: MC_CGM_SC_DC0[DIV]=3h.
4. Configure DRUN to turn off PLL1 (to save power), turn off PLL0 so PLL0 can be re-configured for the required 200MHz self-test clock frequency, turn on the XOSC as it is needed to provide the PLL0 reference clock during self-test, and select IRCOSC to source SYS_CLK:
    a. Select IRCOSC as the SYS_CLK clock source: MC_ME_DRUN_MC[SYSCLK]=0000b.
    b. Enable XOSC: MC_ME_DRUN_MC[XOSCON]=1 (will be required for PLL0).
    c. Disable PLL0: MC_ME_DRUN_MC[PLL0ON]=0.
    d. Disable PLL1: MC_ME_DRUN_MC[PLL1ON]=0.
    e. Gate off all peripheral clocks because they should not be running: MC_ME_RUN_PC0[DRUN]=0, MC_ME_PCTLn[RUN_CFG]=000b.

5. Activate new configuration by performing mode change to DRUN: MC_ME_MCTL[TARGET_MODE]=0011b (see the MC_ME chapter for mode change request details).
6. Configure PLL0 for required 200MHz self-test frequency:
   a. Configure PLL0 to provide 200MHz at its PHI output: PLLDIG_PLL0DV (values depend on XOSC frequency).
   b. Select XOSC as PLL0's reference clock: MC_CGM_AC3_SC[SELCTL]=1.
7. Configure DRUN mode to turn on PLL0 and select PLL0 PHI to source SYS_CLK:
   a. Enable XOSC: MC_ME_DRUN_MC[XOSCON]=1.
   b. Enable PLL0: MC_ME_DRUN_MC[PLL0ON]=1.
   c. Disable PLL1: MC_ME_DRUN_MC[PLL1ON]=0.
   d. Select PLL0 PHI as the SYS_CLK clock source: MC_ME_DRUN_MC[SYSCLK]=0010b.
8. Activate new configuration by performing mode change to DRUN: MC_ME_MCTL[TARGET_MODE]=0011b (see the MC_ME chapter for mode change request details).
9. Select PLL0 PHI to source all auxiliary clocks used during self-test:
   a. Select PLL0 PHI as the MOTC_CLK, SGEN_CLK, and ADC_CLK clock source: MC_CGM_AC0_SC[SELCTL]=2h.
   b. Select PLL0 PHI as the LFAST PLL clock source: MC_CGM_AC5_SC[SELCTL]=2h.

70.6.3.2 Shutdown self-test initiation procedure

To initiate a shutdown self-test, execute the following steps for the desired self-test mode. All accesses must be processed without interruption. The values to write to each register appear in Table 70-4.

1. Configure the STCU_SKC register by first writing the key1 value and then writing the key2 value.
2. Configure the STCU_MBn_CTRL registers.
3. Configure the STCU_CFG register.

4. Configure all STCU_LBn_CTRL registers.
5. Configure all STCU_LBn_PCS registers.
6. Configure all STCU_LBn_MISRELSW registers.
7. Configure all STCU_LBn_MISREHSW registers.
8. Configure the STCU_WDG register.
9. Configure the STCU_LBRMSW register.
10. Configure the DCL_IPS0 register.
11. Configure the STCU_RUNSW register. This access immediately starts the shutdown self-test execution.

NOTE

During shutdown self-tests, once the STCU2 registers are configured they cannot be overridden via IPS until the self-testing is complete.

The following table identifies the values to write to the STCU registers to configure each type of shutdown self-test.

Table 70-4. STCU register configuration for shutdown self-tests

STCU register	Shutdown self-tests		
	Normal	Long	Diagnostic
SKC	Key1: 0x753F924E Key2: 0x8AC06DB1		
MB0_CTRL	0x91000000		
MB1_CTRL	0x98000000		
MB2_CTRL	0x93000000		
MB3_CTRL	0x94000000		
MB4_CTRL	0x95000000		
MB5_CTRL	0x96000000		
MB6_CTRL	0x97000000		
MB7_CTRL	0x10000000		
MB8_CTRL	0x99000000		
MB9_CTRL	0x9A000000		
MB10_CTRL	0x9B000000		
MB11_CTRL	0x9C000000		
MB12_CTRL	0x9D000000		
MB13_CTRL	0x9E000000		
MB14_CTRL	0x9F000000		

Table continues on the next page...

Table 70-4. STCU register configuration for shutdown self-tests (continued)

STCU register	Shutdown self-tests		
	Normal	Long	Diagnostic
MB15_CTRL	0xA0000000		
MB16_CTRL	0xA1000000		
MB17_CTRL	0xA2000000		
MB18_CTRL	0xA3000000		
MB19_CTRL	0xA4000000		
MB20_CTRL	0xA5000000		
MB21_CTRL	0xA6000000		
MB22_CTRL	0xA7000000		
MB23_CTRL	0xA8000000		
MB24_CTRL	0xA9000000		
MB25_CTRL	0xAA000000		
MB26_CTRL	0x00000000		
CFG	0x12000008	0x12000000	0x12000010
LB0_CTRL	0x83031107		
LB1_CTRL	0x82031107		
LB2_CTRL	0x7F031107		
LB3_CTRL	0x01031107		
LB0_PCS	0x00006978		
LB1_PCS	0x00000708		
LB2_PCS	0x00000B00		
LB3_PCS	0x00000740		
LB0_MISRELSW	0x6D73AE0D		
LB1_MISRELSW	0xF1B392F7		
LB2_MISRELSW	0xC97B3FA7		
LB3_MISRELSW	0x5D01B854		
LB0_MISREHSW	0xBD380512		
LB1_MISREHSW	0x18916197		
LB2_MISREHSW	0xFA8FB16E		
LB3_MISREHSW	0x10806935		
WDG	0x00060000	0x00070000	0x00080000
LBRMSW	0x0000000F		
DCL_IPS0	0x03008214		
RUNSW	0x00000301		

70.6.3.3 Shutdown self-test execution sequence

1. The user configures the STCU by following the self-test initiation procedure.

2. The STCU powers a self-test setup phase.
3. Self-test executes / MBIST phase
4. Self-test executes / LBIST phase
5. The STCU collects all BIST results and stores them internally.
6. The STCU arranges a long functional reset.

70.7 MBIST partitioning

The following table describes the mapping between the MBIST number, as the STCU2 captures it, and the appropriate memory in the design.

Table 70-5. MBIST mapping

MBIST number	Memory in design
MBIST0	BAM ROM
MBIST1	DMA RAM
MBIST2	System RAM 1
MBIST3	System RAM 0
MBIST4	System RAM 3
MBIST5	System RAM 2
MBIST6	System RAM 5
MBIST7	System RAM 4
MBIST8	DLMEM 1
MBIST9	DLMEM 0
MBIST10	iCache Data Memory 3
MBIST11	iCache Data Memory 2
MBIST12	iCache Data Memory 1
MBIST13	iCache Data Memory 0
MBIST14	iCache Tag Memory
MBIST15	dCache Data Memory 3
MBIST16	dCache Data Memory 2
MBIST17	dCache Data Memory 1
MBIST18	dCache Data Memory 0
MBIST19	dCache Tag Memory
MBIST20	FlexCAN RAM 2
MBIST21	FlexCAN RAM 1
MBIST22	FlexCAN RAM 0
MBIST23	FlexRay LRAM 1

Table continues on the next page...

Table 70-5. MBIST mapping (continued)

MBIST number	Memory in design
MBIST24	FlexRay LRAM 0
MBIST25	FlexRay DRAM
MBIST26	ENET Memory

70.8 LBIST partitioning

The following table describes the mapping among:

- the LBIST number, as the STCU captures it
- the device partition corresponding to the number
- all modules that take part in the appropriate LBIST

Table 70-6. LBIST mapping

LBIST number	Device partition	Modules participating in LBIST
LBIST0	Partition C0	z4 Main Core_0 PFLASHC Flash memory NPC NXMC0 and NXMC1 MEMU XBAR DSMC DMA EIM INTC NAL NAP PRAMC SWT SMPU STM
LBIST1	Partition C1	z4 Checker Core_0s CMU1 RCCU
LBIST2	Partition P0	DMA_CH_MUX0 FlexRay WKPU

Table continues on the next page...

Table 70-6. LBIST mapping (continued)

LBIST number	Device partition	Modules participating in LBIST
		MC_ME MC_PCU SIUL2 IO-muxing logic CRC PIT DSPI0 and DSPI1 LIN1 FlexPWM1 eTimer1 CTU1 SENT0 SIPI LFAST BAM FlexCAN0, FlexCAN1, and FlexCAN2 ADC1 and ADC3 ENET
LBIST3	Partition P1	DMA_CH_MUX1 FCCU FOSU CMU0, CMU2, CMU3, and CMU4 DSPI2 and DSPI3 SGEN LIN0 FlexPWM0 eTimer0 and eTimer2 CTU0 SENT1 ADC0 and ADC2

Chapter 71
Register Protection (REG_PROT)

71.1 Overview

NOTE

> For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

The protection module is located between the module under protection and the peripheral bridge (PBRIDGE/AIPS-Lite). This is shown in the following figure.

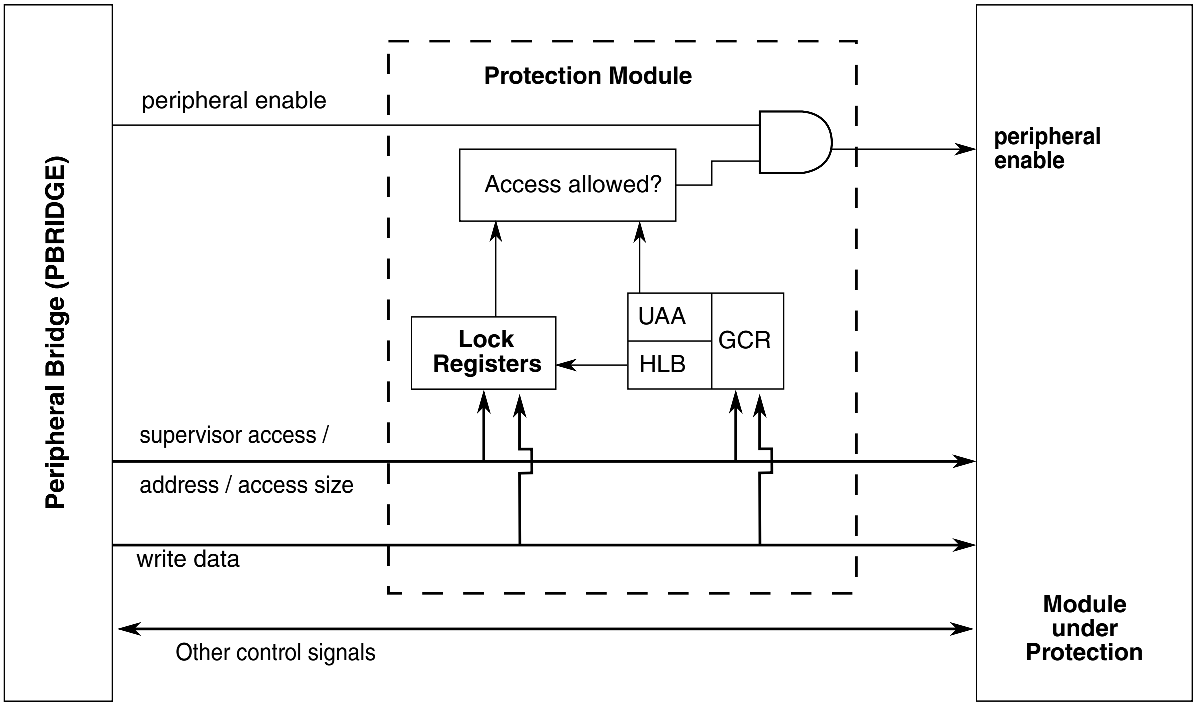


Figure 71-1. REG_PROT Block Diagram

71.2 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

71.3 Modes of operation

The Register Protection module is operable when the module under protection is operable.

71.4 External signal description

There are no external signals.

71.5 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in the following figure.

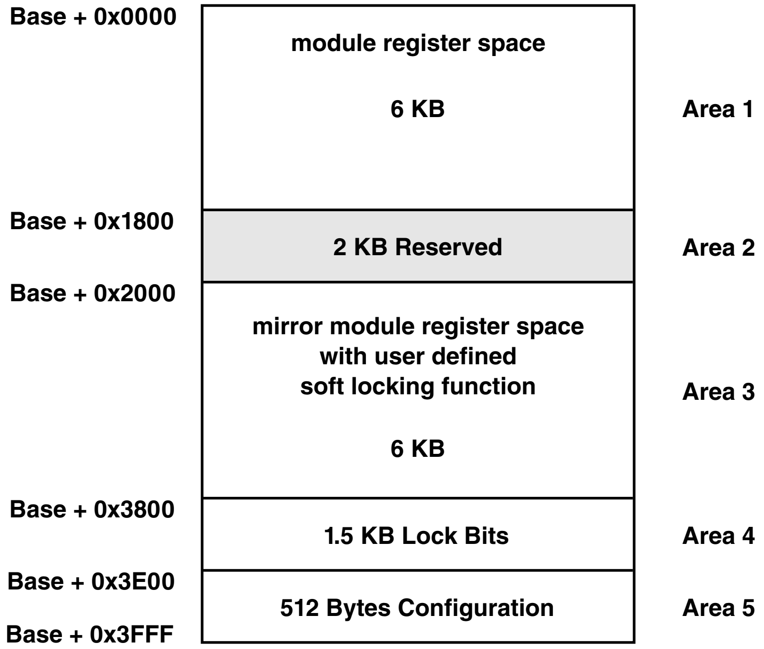


Figure 71-2. REG_PROT Memory Diagram

- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an automatic write and lock operation. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.
- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

71.5.1 Memory map

Following is the memory map for a module which is protected via a REG_PROT module.

Table 71-1. Generic Protected Module Memory Map

Offset	Use
0x0000	Module Register 0 (MR0)
0x0001	Module Register 1 (MR1)
0x0002	Module Register 2 (MR2)
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)
0x1800 - 0x1FFF	Reserved
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 -Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143
0x3E00 - 0x3FFB	Reserved
0x3FFC	Global Configuration Register (GCR)

Note

Reserved registers in area #1 will be handled as specified in the protected module's documentation.

71.5.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

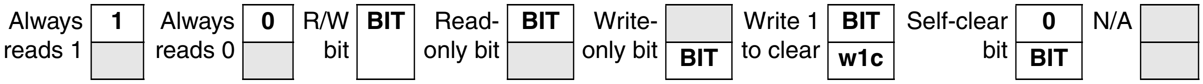


Figure 71-3. Register description

71.5.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

71.5.2.2 Module register and set soft lock bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBRn.SLBm, according to the mapping described in the Soft Lock Bit Register section.

71.6 Memory map and registers

REG_PROT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3800	Soft Lock Bit Register n (REG_PROT_SLBRn)	8	R/W	00h	71.6.1/2825
3FFC	Global Configuration Register (REG_PROT_GCR)	32	R/W	0000_0000h	71.6.2/2827

71.6.1 Soft Lock Bit Register *n* (REG_PROT_SLBRn)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each Soft Lock Bit also has a corresponding Write Enable bit in the same register that controls whether the Soft Lock Bit can be written. The following table shows the mapping between the Soft Lock Bits to the bytes in memory area #1.

Table 71-2. Soft Lock Bits vs. Protected Address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6

Table continues on the next page...

Table 71-2. Soft Lock Bits vs. Protected Address (continued)

Soft Lock Bit	Protected address
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

NOTE

- As many as 1536 Soft Lock Bit Registers (SLBRn) may be implemented, depending on the number of protected module register bytes.
- Access in User Mode is read-only; in Supervisor Mode access is read/write.

Address: 0h base + 3800h offset = 3800h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0	0	SLB0	SLB1	SLB2	SLB3
Write	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

REG_PROT_SLBRn field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB) WE0 enables writing to SLB0 1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB) WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified
2 WE2	Write Enable Bits for Soft Lock Bits (SLB) WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB) WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MRn register

Table continues on the next page...

REG_PROT_SLBRn field descriptions (continued)

Field	Description
	SLB0 can block accesses to MR[n *4 + 0] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
5 SLB1	Soft Lock Bits for one MRn register SLB1 can block accesses to MR[n *4 + 1] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
6 SLB2	Soft Lock Bits for one MRn register SLB2 can block accesses to MR[n *4 + 2] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
7 SLB3	Soft Lock Bits for one MRn register SLB3 can block accesses to MR[n *4 + 3] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable

71.6.2 Global Configuration Register (REG_PROT_GCR)

NOTE

Access in User Mode is read-only; in Supervisor Mode access is read/write.

Address: 0h base + 3FFCh offset = 3FFCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HLB	0							UAA	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REG_PROT_GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit This register cannot be cleared once it is set by software. It can only be cleared by a system reset.

Table continues on the next page...

REG_PROT_GCR field descriptions (continued)

Field	Description
	NOTE: Access in User Mode is read-only; in Supervisor Mode access is read/write. 1 All SLB bits are write protected and can not be modified. 0 All SLB bits are accessible and can be modified.
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 UAA	User Access Allowed. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

71.7 Functional description

The following sections describe the functional characteristics of the Register Protection module.

71.7.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size depends on the SoC and/or module.

For all addresses (peripheral registers) that are protected there are SLBRn.SLBm bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBRn.SLBm bit is always 0b0 no matter what software is writing to.

71.7.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBRn.SLBm bit needs to be changed. This can be done using the following methods:

- Modify the SLBRn.SLBm directly by writing to area #4
- Set the SLBRn.SLBm bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

71.7.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBRn.SLBm bit has a mask bit SLBRn.WEm which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

The following figure shows two modification examples. In the left example there is a write access to the SLBRn register specifying a mask value which allows modification of all SLBRn.SLBm bits. The example on the right specifies a mask which only allows modification of the bits SLBRn.SLB[3:1].

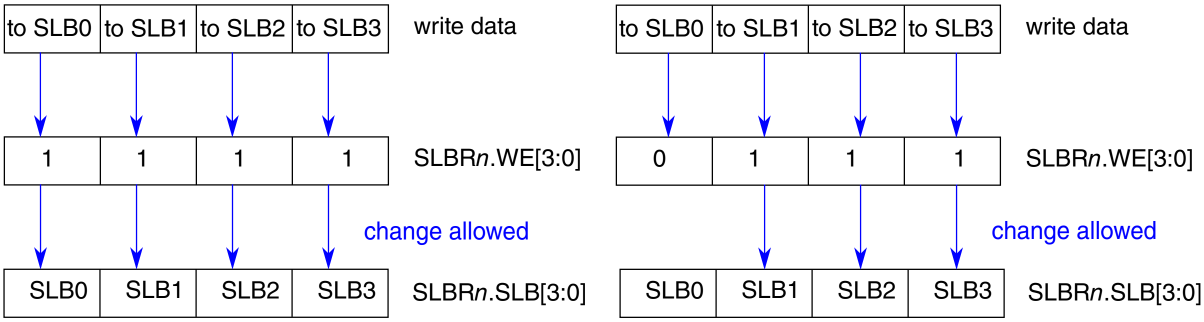


Figure 71-4. Change lock settings directly via Area #4

The previous figure shows four registers that can be protected 8-bit wise. The following figures show registers with 16- and 32-bit protection respectively:

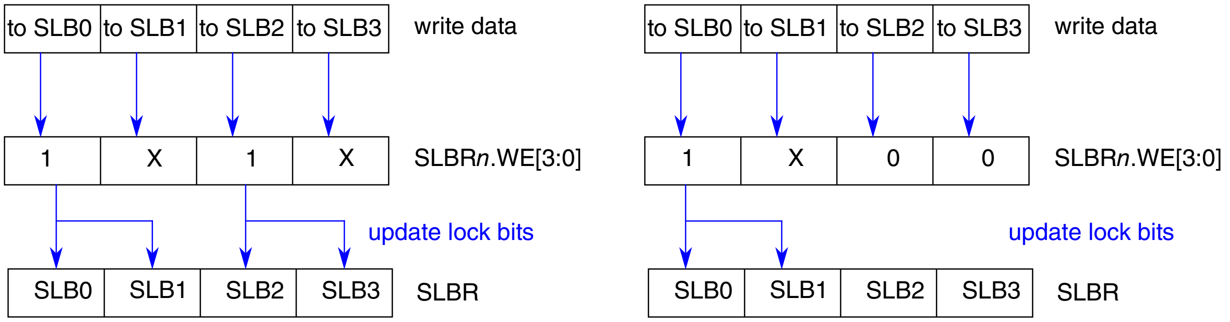


Figure 71-5. Change lock settings for 16-bit protected addresses

On the right side of the previous figure it is shown that the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[1] also. This is done as the address reflected by SLBRn.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBRn.WE[0] must be set while SLBRn.WE[1] does not matter. As the enable bits SLBRn.WE[3:2] are cleared the lock bits SLBRn.SLB[3:2] remain unchanged.

In the example on the left side of the previous figure the data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] and the data written to SLBRn.SLB[2] is mirrored to SLBRn.SLB[3] as for both registers the write enables are set.

The next figure shows a 32-bit wise protected register. When SLBRn.WE[0] is set the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[3:1] also. Otherwise SLBRn.SLB[3:0] remains unchanged.

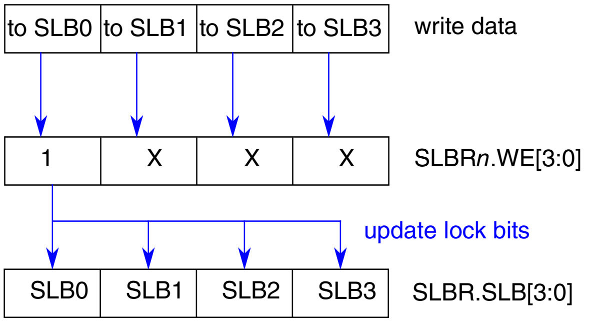


Figure 71-6. Change lock settings for 32-bit protected addresses

The following figure shows a mixed protection size configuration:

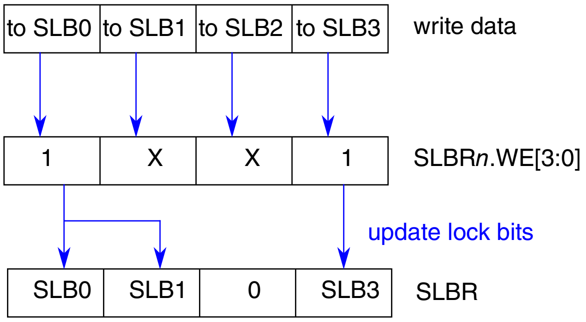


Figure 71-7. Change lock settings for mixed protection

The data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBRn.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBRn.SLB[3] is written to SLBRn.SLB[3].

71.7.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. For example:

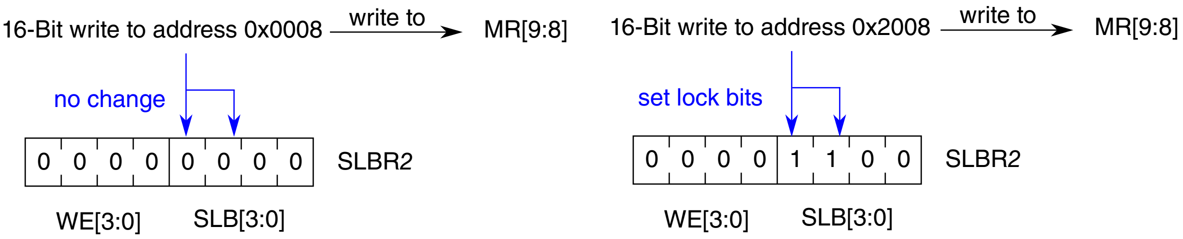


Figure 71-8. Enable Locking Via Mirror Module Space (Area #3)

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (see the left part of Figure 71-5).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of Figure 71-5).

The following figure shows an example where some addresses are protected and some are not:

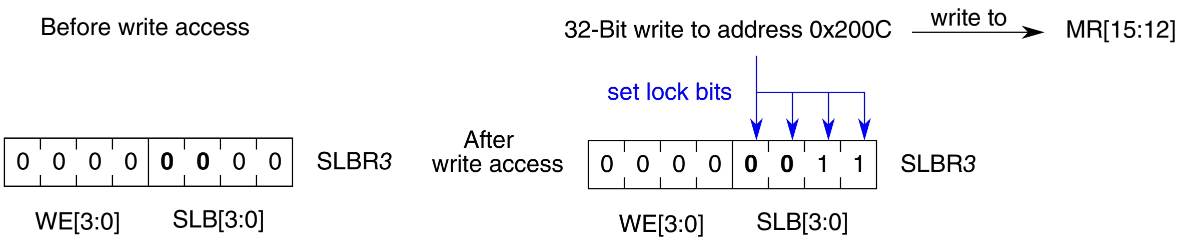


Figure 71-9. Enable locking for protected and unprotected addresses

In the previous figure addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note

> Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

71.7.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in Change lock settings directly via Area #4 and Enable locking via mirror module space (Area #3) is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there is a system reset.

71.7.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to Figure 71-2

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.

6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set

Chapter 72
Wakeup Unit (WKPU)

72.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Wakeup Unit (WKPU) supports one external source that can cause non-maskable interrupts to on-chip cores and wakeup events to the system. The figure below shows the WKPU block diagram and its interfaces to other system components.

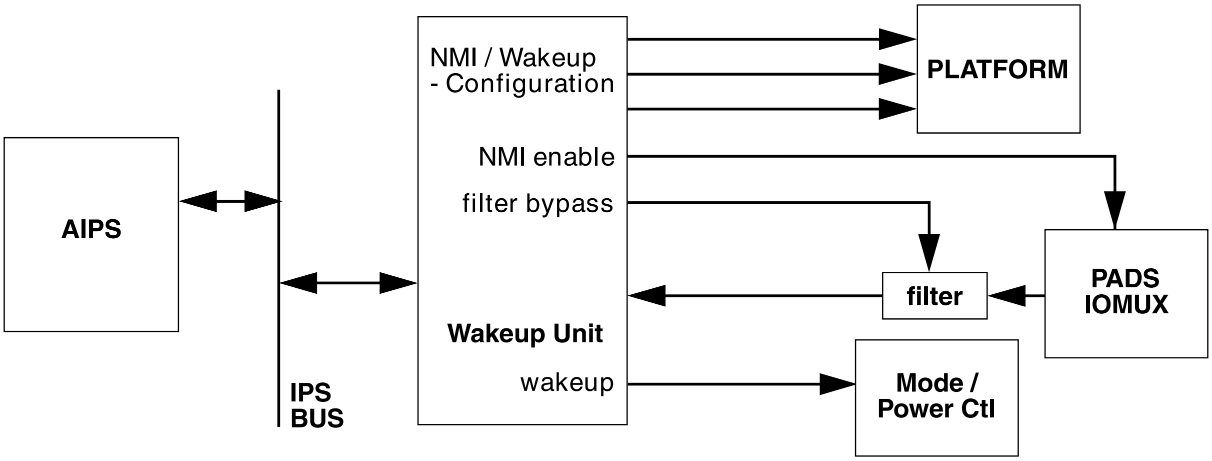


Figure 72-1. WKPU and connected system components

72.2 Features

The WKPU supports these features:

- Non-maskable Interrupt support with:

  - One external NMI source
  - One analog glitch filter
- Independent interrupt destination for each core:
  - Non-maskable interrupt
  - Critical interrupt
  - Machine check request
- Active edge selection control for events to each core
- Configurable system wakeup triggering from NMI source(s)

72.3 External signal description

The WKPU has signal inputs that can be used as external interrupt sources in normal run mode or as system wakeup sources in certain power down modes.

72.4 WKPU memory map and register definition

This section provides a detailed description of all registers accessible in the WKPU module.

NOTE

Reserved registers will read as 0, writes will have no effect. Transfer error will be generated when trying to access completely reserved register space.

WKPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	NMI Status Flag Register (WKPU_NSR)	32	w1c	0000_0000h	72.4.1/2836
8	NMI Configuration Register (WKPU_NCR)	32	R/W	See section	72.4.2/2838

72.4.1 NMI Status Flag Register (WKPU_NSR)

This register holds the non-maskable interrupt status flags.

NOTE

This register is accessible by 8-, 16-, and 32-bit read/write operations.

Access: User read/write

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NIF0	NOVF0	0						0		0					
W	w1c	w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		0						0		0					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_NSR field descriptions

Field	Description
0 NIF0	NMI Status Flag 0. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE0 or NFEE0 set), NIF0 causes an interrupt request. 0 No event has occurred on the pad 1 An event as defined by NREE0 and NFEE0 has occurred
1 NOVF0	NMI Overrun Status Flag 0. This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF0 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE0 or NFEE0 set), NOVF0 causes an interrupt request. 0 No overrun has occurred on NMI input 0 1 An overrun has occurred on NMI input 0
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

WKPU_NSR field descriptions (continued)

Field	Description
18–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

72.4.2 NMI Configuration Register (WKPU_NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

NOTE

- This register is accessible by 8-, 16-, and 32-bit read/write operations.
- Writing a 0 to both NREE[*n*] and NFEE[*n*] disables the NMI functionality completely (i.e., no system wakeup or interrupt will be generated on any pad activity).

Access: User read/write

Address: 0h base + 8h offset = 8h

<table>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>R</td><td rowspan="2">NLOCK0</td><td colspan="2" rowspan="2">NDSS0</td><td rowspan="2">NWRE0</td><td>0</td><td rowspan="2">NREE0</td><td rowspan="2">NFEE0</td><td rowspan="2">NFE0</td><td colspan="8" rowspan="2">Reserved</td></tr>
<tr><td>W</td><td>▨</td></tr>
<tr><td>Reset</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td></tr>
</table>

<table>
<tr><td>Bit</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>R</td><td colspan="16" rowspan="2">Reserved</td></tr>
<tr><td>W</td></tr>
<tr><td>Reset</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td><td>0*</td></tr>
</table>

* Notes:

- See the chip-specific WKPU information for the reset value of this register.

WKPU_NCR field descriptions

Field	Description
0 NLOCK0	NMI Configuration Lock Register 0. Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset . Writing a 0 has no effect.

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
1–2 NDSS0	NMI Destination Source Select 0. **NOTE:** If this field is configured for machine check exception, the following steps must be performed in order to receive the exception on the NMI Pad: 1. Set MSR[ME], Machine Check Enable field of the core's Machine State Register, to 1. 2. Set HID0[EMCP] to 1. 00 Non-maskable interrupt 01 Critical interrupt 10 Machine check request 11 Reserved
3 NWRE0	NMI Wakeup Request Enable 0 **NOTE:** If wakeup requests are disabled, the corresponding NDSS field must be set to 11 to disable wakeups from an NMI, critical interrupt, or machine check. 0 System wakeup requests from the corresponding NIF0 bit are disabled. 1 A set NIF0 bit or set NOVF0 bit causes a system wakeup request.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 NREE0	NMI Rising-edge Events Enable 0. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
6 NFEE0	NMI Falling-edge Events Enable 0. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
7 NFE0	NMI Filter Enable 0. Enable analog glitch filter on the NMI pad input. 0 Filter is disabled 1 Filter is enabled
8–31 Reserved	This field is reserved. Always write 0 to this field.

72.5 Functional description

This section provides a functional description of the WKPU.

72.5.1 Non-maskable interrupts

The WKPU supports the generation of 3 types of interrupts per NMI input to the SoC. The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event, though it creates an overrun condition.

Each NMI passes through a bypassable analog glitch filter.

NOTE

Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

NOTE

The figure below represents a generic configuration and might not represent this particular device's configuration. See the chip-specific information for details on this chip's WKPU.

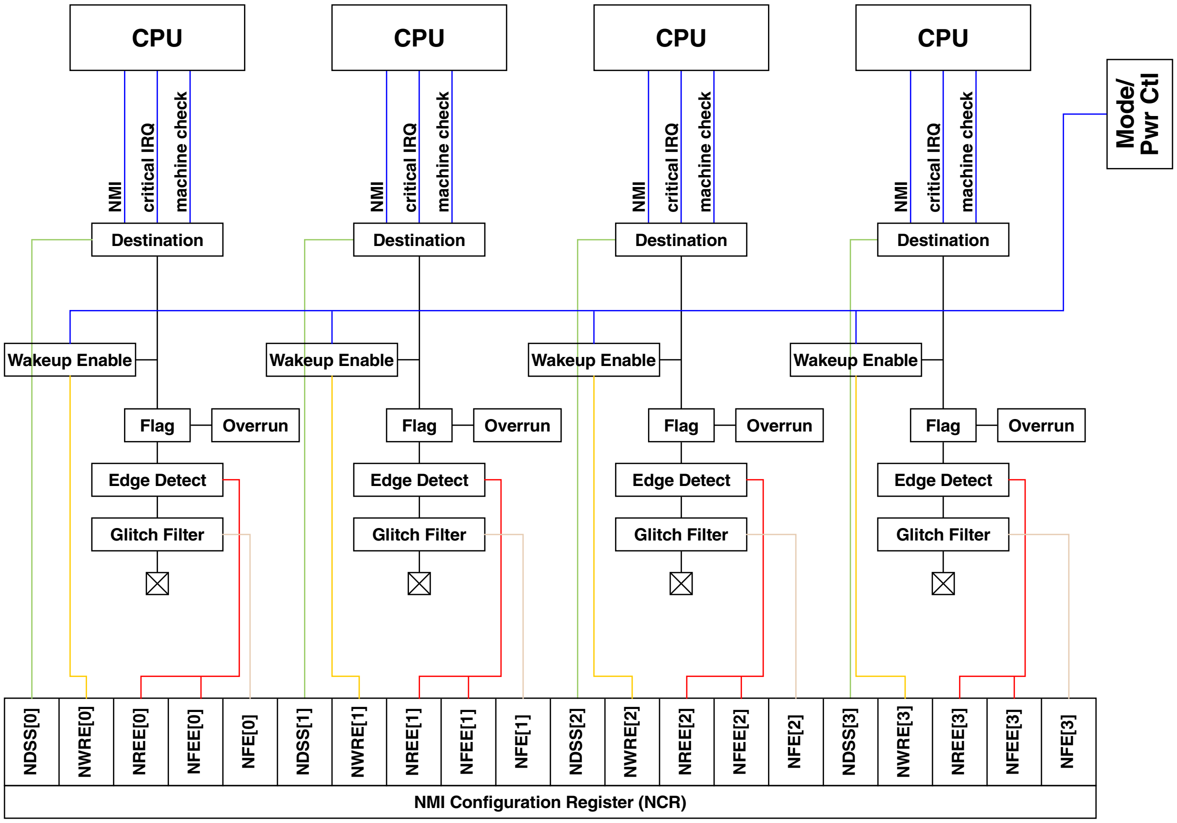


Figure 72-2. NMI pad diagram

72.5.1.1 NMI management

Each NMI can be enabled or disabled independently. This can be performed using the single NCR register laid out to contain all configuration bits for a given NMI in a single byte (see NMI Configuration Register (WKPU_NCR)). A pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

Note

> After reset, NREE and NFEE are set to '0', therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once a pad's NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI. Please see chip specific section for details on NMI implementation.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See NMI Configuration Register (WKPU_NCR) for details.

Each NMI supports a status flag and an overrun flag which are located in the NSR register (see NMI Status Flag Register (WKPU_NSR)). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (i.e. has not yet been cleared).

Note

> The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.
>
> During an NMI ISR, on wakeup of the SoC from an NMI, any writes to ECC-protected memory must have the correct ECC.

Appendix A Protected Registers

The following table identifies registers that are protected on this device.

Table A-1. Protected registers

Module	Register	Size	Offset	Address	Protect size
ADC0	MCR	32	0x0	0xFBE00000	32-bit
ADC0	IMR	32	0x20	0xFBE00020	32-bit
ADC0	CIMR0	32	0x24	0xFBE00024	32-bit
ADC0	WTIMR	32	0x34	0xFBE00034	32-bit
ADC0	DMAE	32	0x40	0xFBE00040	32-bit
ADC0	DMAR0	32	0x44	0xFBE00044	32-bit
ADC0	THRHLR0	32	0x60	0xFBE00060	32-bit
ADC0	THRHLR1	32	0x64	0xFBE00064	32-bit
ADC0	THRHLR2	32	0x68	0xFBE00068	32-bit
ADC0	THRHLR3	32	0x6C	0xFBE0006C	32-bit
ADC0	PSCR	32	0x80	0xFBE00080	32-bit
ADC0	PSR0	32	0x84	0xFBE00084	32-bit
ADC0	CTR0	32	0x94	0xFBE00094	32-bit
ADC0	NCMR0	32	0xA4	0xFBE000A4	32-bit
ADC0	JCMR0	32	0xB4	0xFBE000B4	32-bit
ADC0	PDEDR	32	0xC8	0xFBE000C8	32-bit
ADC0	THRHLR4	32	0x280	0xFBE00280	32-bit
ADC0	THRHLR5	32	0x284	0xFBE00284	32-bit
ADC0	THRHLR6	32	0x288	0xFBE00288	32-bit
ADC0	THRHLR7	32	0x28C	0xFBE0028C	32-bit
ADC0	THRHLR8	32	0x290	0xFBE00290	32-bit
ADC0	THRHLR9	32	0x294	0xFBE00294	32-bit
ADC0	THRHLR10	32	0x298	0xFBE00298	32-bit
ADC0	THRHLR11	32	0x29C	0xFBE0029C	32-bit
ADC0	THRHLR12	32	0x2A0	0xFBE002A0	32-bit
ADC0	THRHLR13	32	0x2A4	0xFBE002A4	32-bit
ADC0	THRHLR14	32	0x2A8	0xFBE002A8	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
ADC0	THRHLR15	32	0x2AC	0xFBE002AC	32-bit
ADC0	CWSELR0	32	0x2B0	0xFBE002B0	32-bit
ADC0	CWSELR1	32	0x2B4	0xFBE002B4	32-bit
ADC0	CWENR0	32	0x2E0	0xFBE002E0	32-bit
ADC0	AWORR0	32	0x2F0	0xFBE002F0	32-bit
ADC0	STCR1	32	0x340	0xFBE00340	32-bit
ADC0	STCR2	32	0x344	0xFBE00344	32-bit
ADC0	STCR3	32	0x348	0xFBE00348	32-bit
ADC0	STBRR	32	0x34C	0xFBE0034C	32-bit
ADC0	STAW0R	32	0x380	0xFBE00380	32-bit
ADC0	STAW1AR	32	0x384	0xFBE00384	32-bit
ADC0	STAW1BR	32	0x388	0xFBE00388	32-bit
ADC0	STAW2R	32	0x38C	0xFBE0038C	32-bit
ADC0	STAW4R	32	0x394	0xFBE00394	32-bit
ADC0	STAW5R	32	0x398	0xFBE00398	32-bit
ADC0	CALBISTREG	32	0x3A0	0xFBE003A0	32-bit
ADC0	OFSGNUSR	32	0x3A8	0xFBE003A8	32-bit
ADC1	MCR	32	0x0	0xFFE04000	32-bit
ADC1	IMR	32	0x20	0xFFE04020	32-bit
ADC1	CIMR0	32	0x24	0xFFE04024	32-bit
ADC1	WTIMR	32	0x34	0xFFE04034	32-bit
ADC1	DMAE	32	0x40	0xFFE04040	32-bit
ADC1	DMAR0	32	0x44	0xFFE04044	32-bit
ADC1	THRHLR0	32	0x60	0xFFE04060	32-bit
ADC1	THRHLR1	32	0x64	0xFFE04064	32-bit
ADC1	THRHLR2	32	0x68	0xFFE04068	32-bit
ADC1	THRHLR3	32	0x6C	0xFFE0406C	32-bit
ADC1	PSCR	32	0x80	0xFFE04080	32-bit
ADC1	PSR0	32	0x84	0xFFE04084	32-bit
ADC1	CTR0	32	0x94	0xFFE04094	32-bit
ADC1	NCMR0	32	0xA4	0xFFE040A4	32-bit
ADC1	JCMR0	32	0xB4	0xFFE040B4	32-bit
ADC1	PDEDR	32	0xC8	0xFFE040C8	32-bit
ADC1	THRHLR4	32	0x280	0xFFE04280	32-bit
ADC1	THRHLR5	32	0x284	0xFFE04284	32-bit
ADC1	THRHLR6	32	0x288	0xFFE04288	32-bit
ADC1	THRHLR7	32	0x28C	0xFFE0428C	32-bit
ADC1	THRHLR8	32	0x290	0xFFE04290	32-bit
ADC1	THRHLR9	32	0x294	0xFFE04294	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
ADC1	THRHLR10	32	0x298	0xFFE04298	32-bit
ADC1	THRHLR11	32	0x29C	0xFFE0429C	32-bit
ADC1	THRHLR12	32	0x2A0	0xFFE042A0	32-bit
ADC1	THRHLR13	32	0x2A4	0xFFE042A4	32-bit
ADC1	THRHLR14	32	0x2A8	0xFFE042A8	32-bit
ADC1	THRHLR15	32	0x2AC	0xFFE042AC	32-bit
ADC1	CWSELR0	32	0x2B0	0xFFE042B0	32-bit
ADC1	CWSELR1	32	0x2B4	0xFFE042B4	32-bit
ADC1	CWENR0	32	0x2E0	0xFFE042E0	32-bit
ADC1	AWORR0	32	0x2F0	0xFFE042F0	32-bit
ADC1	STCR1	32	0x340	0xFFE04340	32-bit
ADC1	STCR2	32	0x344	0xFFE04344	32-bit
ADC1	STCR3	32	0x348	0xFFE04348	32-bit
ADC1	STBRR	32	0x34C	0xFFE0434C	32-bit
ADC1	STAW0R	32	0x380	0xFFE04380	32-bit
ADC1	STAW1AR	32	0x384	0xFFE04384	32-bit
ADC1	STAW1BR	32	0x388	0xFFE04388	32-bit
ADC1	STAW2R	32	0x38C	0xFFE0438C	32-bit
ADC1	STAW4R	32	0x394	0xFFE04394	32-bit
ADC1	STAW5R	32	0x398	0xFFE04398	32-bit
ADC1	CALBISTREG	32	0x3A0	0xFFE043A0	32-bit
ADC1	OFSGNUSR	32	0x3A8	0xFFE043A8	32-bit
ADC2	MCR	32	0x0	0xFBE08000	32-bit
ADC2	IMR	32	0x20	0xFBE08020	32-bit
ADC2	CIMR0	32	0x24	0xFBE08024	32-bit
ADC2	WTIMR	32	0x34	0xFBE08034	32-bit
ADC2	DMAE	32	0x40	0xFBE08040	32-bit
ADC2	DMAR0	32	0x44	0xFBE08044	32-bit
ADC2	THRHLR0	32	0x60	0xFBE08060	32-bit
ADC2	THRHLR1	32	0x64	0xFBE08064	32-bit
ADC2	THRHLR2	32	0x68	0xFBE08068	32-bit
ADC2	THRHLR3	32	0x6C	0xFBE0806C	32-bit
ADC2	PSCR	32	0x80	0xFBE08080	32-bit
ADC2	PSR0	32	0x84	0xFBE08084	32-bit
ADC2	CTR0	32	0x94	0xFBE08094	32-bit
ADC2	NCMR0	32	0xA4	0xFBE080A4	32-bit
ADC2	JCMR0	32	0xB4	0xFBE080B4	32-bit
ADC2	PDEDR	32	0xC8	0xFBE080C8	32-bit
ADC2	THRHLR4	32	0x280	0xFBE08280	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
ADC2	THRHLR5	32	0x284	0xFBE08284	32-bit
ADC2	THRHLR6	32	0x288	0xFBE08288	32-bit
ADC2	THRHLR7	32	0x28C	0xFBE0828C	32-bit
ADC2	THRHLR8	32	0x290	0xFBE08290	32-bit
ADC2	THRHLR9	32	0x294	0xFBE08294	32-bit
ADC2	THRHLR10	32	0x298	0xFBE08298	32-bit
ADC2	THRHLR11	32	0x29C	0xFBE0829C	32-bit
ADC2	THRHLR12	32	0x2A0	0xFBE082A0	32-bit
ADC2	THRHLR13	32	0x2A4	0xFBE082A4	32-bit
ADC2	THRHLR14	32	0x2A8	0xFBE082A8	32-bit
ADC2	THRHLR15	32	0x2AC	0xFBE082AC	32-bit
ADC2	CWSELR0	32	0x2B0	0xFBE082B0	32-bit
ADC2	CWSELR1	32	0x2B4	0xFBE082B4	32-bit
ADC2	CWENR0	32	0x2E0	0xFBE082E0	32-bit
ADC2	AWORR0	32	0x2F0	0xFBE082F0	32-bit
ADC2	STCR1	32	0x340	0xFBE08340	32-bit
ADC2	STCR2	32	0x344	0xFBE08344	32-bit
ADC2	STCR3	32	0x348	0xFBE08348	32-bit
ADC2	STBRR	32	0x34C	0xFBE0834C	32-bit
ADC2	STAW0R	32	0x380	0xFBE08380	32-bit
ADC2	STAW1AR	32	0x384	0xFBE08384	32-bit
ADC2	STAW1BR	32	0x388	0xFBE08388	32-bit
ADC2	STAW2R	32	0x38C	0xFBE0838C	32-bit
ADC2	STAW4R	32	0x394	0xFBE08394	32-bit
ADC2	STAW5R	32	0x398	0xFBE08398	32-bit
ADC2	CALBISTREG	32	0x3A0	0xFBE083A0	32-bit
ADC2	OFSGNUSR	32	0x3A8	0xFBE083A8	32-bit
ADC3	MCR	32	0x0	0xFFE0C000	32-bit
ADC3	IMR	32	0x20	0xFFE0C020	32-bit
ADC3	CIMR0	32	0x24	0xFFE0C024	32-bit
ADC3	WTIMR	32	0x34	0xFFE0C034	32-bit
ADC3	DMAE	32	0x40	0xFFE0C040	32-bit
ADC3	DMAR0	32	0x44	0xFFE0C044	32-bit
ADC3	THRHLR0	32	0x60	0xFFE0C060	32-bit
ADC3	THRHLR1	32	0x64	0xFFE0C064	32-bit
ADC3	THRHLR2	32	0x68	0xFFE0C068	32-bit
ADC3	THRHLR3	32	0x6C	0xFFE0C06C	32-bit
ADC3	PSCR	32	0x80	0xFFE0C080	32-bit
ADC3	PSR0	32	0x84	0xFFE0C084	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
ADC3	CTR0	32	0x94	0xFFE0C094	32-bit
ADC3	NCMR0	32	0xA4	0xFFE0C0A4	32-bit
ADC3	JCMR0	32	0xB4	0xFFE0C0B4	32-bit
ADC3	PDEDR	32	0xC8	0xFFE0C0C8	32-bit
ADC3	THRHLR4	32	0x280	0xFFE0C280	32-bit
ADC3	THRHLR5	32	0x284	0xFFE0C284	32-bit
ADC3	THRHLR6	32	0x288	0xFFE0C288	32-bit
ADC3	THRHLR7	32	0x28C	0xFFE0C28C	32-bit
ADC3	THRHLR8	32	0x290	0xFFE0C290	32-bit
ADC3	THRHLR9	32	0x294	0xFFE0C294	32-bit
ADC3	THRHLR10	32	0x298	0xFFE0C298	32-bit
ADC3	THRHLR11	32	0x29C	0xFFE0C29C	32-bit
ADC3	THRHLR12	32	0x2A0	0xFFE0C2A0	32-bit
ADC3	THRHLR13	32	0x2A4	0xFFE0C2A4	32-bit
ADC3	THRHLR14	32	0x2A8	0xFFE0C2A8	32-bit
ADC3	THRHLR15	32	0x2AC	0xFFE0C2AC	32-bit
ADC3	CWSELR0	32	0x2B0	0xFFE0C2B0	32-bit
ADC3	CWSELR1	32	0x2B4	0xFFE0C2B4	32-bit
ADC3	CWENR0	32	0x2E0	0xFFE0C2E0	32-bit
ADC3	AWORR0	32	0x2F0	0xFFE0C2F0	32-bit
ADC3	STCR1	32	0x340	0xFFE0C340	32-bit
ADC3	STCR2	32	0x344	0xFFE0C344	32-bit
ADC3	STCR3	32	0x348	0xFFE0C348	32-bit
ADC3	STBRR	32	0x34C	0xFFE0C34C	32-bit
ADC3	STAW0R	32	0x380	0xFFE0C380	32-bit
ADC3	STAW1AR	32	0x384	0xFFE0C384	32-bit
ADC3	STAW1BR	32	0x388	0xFFE0C388	32-bit
ADC3	STAW2R	32	0x38C	0xFFE0C38C	32-bit
ADC3	STAW4R	32	0x394	0xFFE0C394	32-bit
ADC3	STAW5R	32	0x398	0xFFE0C398	32-bit
ADC3	CALBISTREG	32	0x3A0	0xFFE0C3A0	32-bit
ADC3	OFSGNUSR	32	0x3A8	0xFFE0C3A8	32-bit
CMU0	CSR	32	0x0	0xFBFB0200	32-bit
CMU0	HFREFR	32	0x8	0xFBFB0208	32-bit
CMU0	LFREFR	32	0xC	0xFBFB020C	32-bit
CMU0	MDR	32	0x18	0xFBFB0218	32-bit
CMU1	CSR	32	0x0	0xFBFB0200	32-bit
CMU1	HFREFR	32	0x8	0xFBFB0208	32-bit
CMU1	LFREFR	32	0xC	0xFBFB020C	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
CMU1	MDR	32	0x18	0xFBFB0218	32-bit
CMU2	CSR	32	0x0	0xFBFB0200	32-bit
CMU2	HFREFR	32	0x8	0xFBFB0208	32-bit
CMU2	LFREFR	32	0xC	0xFBFB020C	32-bit
CMU2	MDR	32	0x18	0xFBFB0218	32-bit
CMU3	CSR	32	0x0	0xFBFB0200	32-bit
CMU3	HFREFR	32	0x8	0xFBFB0208	32-bit
CMU3	LFREFR	32	0xC	0xFBFB020C	32-bit
CMU3	MDR	32	0x18	0xFBFB0218	32-bit
CMU4	CSR	32	0x0	0xFBFB0200	32-bit
CMU4	HFREFR	32	0x8	0xFBFB0208	32-bit
CMU4	LFREFR	32	0xC	0xFBFB020C	32-bit
CMU4	MDR	32	0x18	0xFBFB0218	32-bit
CRC	CFG1	32	0x0	0xFFF64000	32-bit
CRC	CFG2	32	0x10	0xFFF64010	32-bit
CRC	CFG3	32	0x20	0xFFF64020	32-bit
CTU0	TGSISR	32	0x0	0xFBC10000	32-bit
CTU0	TGSCR	16	0x4	0xFBC10004	16-bit
CTU0	T0CR	16	0x6	0xFBC10006	16-bit
CTU0	T1CR	16	0x8	0xFBC10008	16-bit
CTU0	T2CR	16	0x000A	0xFBC1000A	16-bit
CTU0	T3CR	16	0x000C	0xFBC1000C	16-bit
CTU0	T4CR	16	0x000E	0xFBC1000E	16-bit
CTU0	T5CR	16	0x10	0xFBC10010	16-bit
CTU0	T6CR	16	0x12	0xFBC10012	16-bit
CTU0	T7CR	16	0x14	0xFBC10014	16-bit
CTU0	TGSCCR	16	0x16	0xFBC10016	16-bit
CTU0	TGSCRR	16	0x18	0xFBC10018	16-bit
CTU0	CLCR1	32	0x001C	0xFBC1001C	32-bit
CTU0	CLCR2	32	0x20	0xFBC10020	32-bit
CTU0	THCR1	32	0x24	0xFBC10024	32-bit
CTU0	THCR2	32	0x28	0xFBC10028	32-bit
CTU0	CLR1	16	0x002C	0xFBC1002C	16-bit
CTU0	CLR2	16	0x002E	0xFBC1002E	16-bit
CTU0	CLR3	16	0x30	0xFBC10030	16-bit
CTU0	CLR4	16	0x32	0xFBC10032	16-bit
CTU0	CLR5	16	0x34	0xFBC10034	16-bit
CTU0	CLR6	16	0x36	0xFBC10036	16-bit
CTU0	CLR7	16	0x38	0xFBC10038	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
CTU0	CLR8	16	0x003A	0xFBC1003A	16-bit
CTU0	CLR9	16	0x003C	0xFBC1003C	16-bit
CTU0	CLR10	16	0x003E	0xFBC1003E	16-bit
CTU0	CLR11	16	0x40	0xFBC10040	16-bit
CTU0	CLR12	16	0x42	0xFBC10042	16-bit
CTU0	CLR13	16	0x44	0xFBC10044	16-bit
CTU0	CLR14	16	0x46	0xFBC10046	16-bit
CTU0	CLR15	16	0x48	0xFBC10048	16-bit
CTU0	CLR16	16	0x004A	0xFBC1004A	16-bit
CTU0	CLR17	16	0x004C	0xFBC1004C	16-bit
CTU0	CLR18	16	0x004E	0xFBC1004E	16-bit
CTU0	CLR19	16	0x50	0xFBC10050	16-bit
CTU0	CLR20	16	0x52	0xFBC10052	16-bit
CTU0	CLR21	16	0x54	0xFBC10054	16-bit
CTU0	CLR22	16	0x56	0xFBC10056	16-bit
CTU0	CLR23	16	0x58	0xFBC10058	16-bit
CTU0	CLR24	16	0x005A	0xFBC1005A	16-bit
CTU0	FDCR	16	0x006C	0xFBC1006C	16-bit
CTU0	FCR	32	0x70	0xFBC10070	32-bit
CTU0	FTH	32	0x74	0xFBC10074	32-bit
CTU0	IR	16	0x00C4	0xFBC100C4	16-bit
CTU0	COTR	16	0x00C6	0xFBC100C6	16-bit
CTU0	CR	16	0x00C8	0xFBC100C8	16-bit
CTU0	DFR	16	0x00CA	0xFBC100CA	16-bit
CTU0	EXPAR	16	0xCC	0xFBC100CC	16-bit
CTU0	EXPBR	16	0xCE	0xFBC100CE	16-bit
CTU0	CNTRNGR	16	0xD0	0xFBC100D0	16-bit
CTU0	LISTCSR	32	0xD4	0xFBC100D4	32-bit
CTU1	TGSISR	32	0x0	0xFBC10000	32-bit
CTU1	TGSCR	16	0x4	0xFBC10004	16-bit
CTU1	T0CR	16	0x6	0xFBC10006	16-bit
CTU1	T1CR	16	0x8	0xFBC10008	16-bit
CTU1	T2CR	16	0x000A	0xFBC1000A	16-bit
CTU1	T3CR	16	0x000C	0xFBC1000C	16-bit
CTU1	T4CR	16	0x000E	0xFBC1000E	16-bit
CTU1	T5CR	16	0x10	0xFBC10010	16-bit
CTU1	T6CR	16	0x12	0xFBC10012	16-bit
CTU1	T7CR	16	0x14	0xFBC10014	16-bit
CTU1	TGSCCR	16	0x16	0xFBC10016	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
CTU1	TGSCRR	16	0x18	0xFBC10018	16-bit
CTU1	CLCR1	32	0x001C	0xFBC1001C	32-bit
CTU1	CLCR2	32	0x20	0xFBC10020	32-bit
CTU1	THCR1	32	0x24	0xFBC10024	32-bit
CTU1	THCR2	32	0x28	0xFBC10028	32-bit
CTU1	CLR1	16	0x002C	0xFBC1002C	16-bit
CTU1	CLR2	16	0x002E	0xFBC1002E	16-bit
CTU1	CLR3	16	0x30	0xFBC10030	16-bit
CTU1	CLR4	16	0x32	0xFBC10032	16-bit
CTU1	CLR5	16	0x34	0xFBC10034	16-bit
CTU1	CLR6	16	0x36	0xFBC10036	16-bit
CTU1	CLR7	16	0x38	0xFBC10038	16-bit
CTU1	CLR8	16	0x003A	0xFBC1003A	16-bit
CTU1	CLR9	16	0x003C	0xFBC1003C	16-bit
CTU1	CLR10	16	0x003E	0xFBC1003E	16-bit
CTU1	CLR11	16	0x40	0xFBC10040	16-bit
CTU1	CLR12	16	0x42	0xFBC10042	16-bit
CTU1	CLR13	16	0x44	0xFBC10044	16-bit
CTU1	CLR14	16	0x46	0xFBC10046	16-bit
CTU1	CLR15	16	0x48	0xFBC10048	16-bit
CTU1	CLR16	16	0x004A	0xFBC1004A	16-bit
CTU1	CLR17	16	0x004C	0xFBC1004C	16-bit
CTU1	CLR18	16	0x004E	0xFBC1004E	16-bit
CTU1	CLR19	16	0x50	0xFBC10050	16-bit
CTU1	CLR20	16	0x52	0xFBC10052	16-bit
CTU1	CLR21	16	0x54	0xFBC10054	16-bit
CTU1	CLR22	16	0x56	0xFBC10056	16-bit
CTU1	CLR23	16	0x58	0xFBC10058	16-bit
CTU1	CLR24	16	0x005A	0xFBC1005A	16-bit
CTU1	FDCR	16	0x006C	0xFBC1006C	16-bit
CTU1	FCR	32	0x70	0xFBC10070	32-bit
CTU1	FTH	32	0x74	0xFBC10074	32-bit
CTU1	IR	16	0x00C4	0xFBC100C4	16-bit
CTU1	COTR	16	0x00C6	0xFBC100C6	16-bit
CTU1	CR	16	0x00C8	0xFBC100C8	16-bit
CTU1	DFR	16	0x00CA	0xFBC100CA	16-bit
CTU1	EXPAR	16	0xCC	0xFBC100CC	16-bit
CTU1	EXPBR	16	0xCE	0xFBC100CE	16-bit
CTU1	CNTRNGR	16	0xD0	0xFBC100D0	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
CTU1	LISTCSR	32	0xD4	0xFBC100D4	32-bit
DMAMUX0	CHCFG0	8	0x0	0xFBF6C000	8-bit
DMAMUX0	CHCFG1	8	0x1	0xFBF6C001	8-bit
DMAMUX0	CHCFG2	8	0x2	0xFBF6C002	8-bit
DMAMUX0	CHCFG3	8	0x3	0xFBF6C003	8-bit
DMAMUX0	CHCFG4	8	0x4	0xFBF6C004	8-bit
DMAMUX0	CHCFG5	8	0x5	0xFBF6C005	8-bit
DMAMUX0	CHCFG6	8	0x6	0xFBF6C006	8-bit
DMAMUX0	CHCFG7	8	0x7	0xFBF6C007	8-bit
DMAMUX0	CHCFG8	8	0x8	0xFBF6C008	8-bit
DMAMUX0	CHCFG9	8	0x9	0xFBF6C009	8-bit
DMAMUX0	CHCFG10	8	0xA	0xFBF6C00A	8-bit
DMAMUX0	CHCFG11	8	0xB	0xFBF6C00B	8-bit
DMAMUX0	CHCFG12	8	0xC	0xFBF6C00C	8-bit
DMAMUX0	CHCFG13	8	0xD	0xFBF6C00D	8-bit
DMAMUX0	CHCFG14	8	0xE	0xFBF6C00E	8-bit
DMAMUX0	CHCFG15	8	0xF	0xFBF6C00F	8-bit
DMAMUX1	CHCFG0	8	0x0	0xFBF6C000	8-bit
DMAMUX1	CHCFG1	8	0x1	0xFBF6C001	8-bit
DMAMUX1	CHCFG2	8	0x2	0xFBF6C002	8-bit
DMAMUX1	CHCFG3	8	0x3	0xFBF6C003	8-bit
DMAMUX1	CHCFG4	8	0x4	0xFBF6C004	8-bit
DMAMUX1	CHCFG5	8	0x5	0xFBF6C005	8-bit
DMAMUX1	CHCFG6	8	0x6	0xFBF6C006	8-bit
DMAMUX1	CHCFG7	8	0x7	0xFBF6C007	8-bit
DMAMUX1	CHCFG8	8	0x8	0xFBF6C008	8-bit
DMAMUX1	CHCFG9	8	0x9	0xFBF6C009	8-bit
DMAMUX1	CHCFG10	8	0xA	0xFBF6C00A	8-bit
DMAMUX1	CHCFG11	8	0xB	0xFBF6C00B	8-bit
DMAMUX1	CHCFG12	8	0xC	0xFBF6C00C	8-bit
DMAMUX1	CHCFG13	8	0xD	0xFBF6C00D	8-bit
DMAMUX1	CHCFG14	8	0xE	0xFBF6C00E	8-bit
DMAMUX1	CHCFG15	8	0xF	0xFBF6C00F	8-bit
SPI0	MCR	32	0x0	0xFBE70000	32-bit
SPI0	TCR	32	0x8	0xFBE70008	32-bit
SPI0	CTAR0	32	0xC	0xFBE7000C	32-bit
SPI0	CTAR1	32	0x10	0xFBE70010	32-bit
SPI0	CTAR2	32	0x14	0xFBE70014	32-bit
SPI0	CTAR3	32	0x18	0xFBE70018	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
SPI0	RSER	32	0x30	0xFBE70030	32-bit
SPI1	MCR	32	0x0	0xFBE70000	32-bit
SPI1	TCR	32	0x8	0xFBE70008	32-bit
SPI1	CTAR0	32	0xC	0xFBE7000C	32-bit
SPI1	CTAR1	32	0x10	0xFBE70010	32-bit
SPI1	CTAR2	32	0x14	0xFBE70014	32-bit
SPI1	CTAR3	32	0x18	0xFBE70018	32-bit
SPI1	RSER	32	0x30	0xFBE70030	32-bit
SPI2	MCR	32	0x0	0xFBE70000	32-bit
SPI2	TCR	32	0x8	0xFBE70008	32-bit
SPI2	CTAR0	32	0xC	0xFBE7000C	32-bit
SPI2	CTAR1	32	0x10	0xFBE70010	32-bit
SPI2	CTAR2	32	0x14	0xFBE70014	32-bit
SPI2	CTAR3	32	0x18	0xFBE70018	32-bit
SPI2	RSER	32	0x30	0xFBE70030	32-bit
SPI2	MCR	32	0x0	0xFBE70000	32-bit
SPI2	TCR	32	0x8	0xFBE70008	32-bit
SPI2	CTAR0	32	0xC	0xFBE7000C	32-bit
SPI2	CTAR1	32	0x10	0xFBE70010	32-bit
SPI2	CTAR2	32	0x14	0xFBE70014	32-bit
SPI2	CTAR3	32	0x18	0xFBE70018	32-bit
SPI2	RSER	32	0x30	0xFBE70030	32-bit
LFAST	MCR	32	0x0	0xFFFD8000	32-bit
LFAST	SCR	32	0x4	0xFFFD8004	32-bit
LFAST	COCR	32	0x8	0xFFFD8008	32-bit
LFAST	TMCR	32	0xc	0xFFFD800C	32-bit
LFAST	ALCR	32	0x10	0xFFFD8010	32-bit
LFAST	RCDCR	32	0x14	0xFFFD8014	32-bit
LFAST	SLCR	32	0x18	0xFFFD8018	32-bit
LFAST	ICR	32	0x1c	0xFFFD801C	32-bit
LFAST	PICR	32	0x20	0xFFFD8020	32-bit
LFAST	RFCR	32	0x2c	0xFFFD802C	32-bit
LFAST	TIER	32	0x30	0xFFFD8030	32-bit
LFAST	RIER	32	0x34	0xFFFD8034	32-bit
LFAST	RIIER	32	0x38	0xFFFD8038	32-bit
LFAST	PLLCR	32	0x3c	0xFFFD803C	32-bit
ENET	EIMR	32	0x8	0xFC0B0008	32-bit
ENET	ECR	32	0x24	0xFC0B0024	32-bit
ENET	MSCR	32	0x44	0xFC0B0044	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
ENET	RCR	32	0x84	0xFC0B0084	32-bit
ENET	TCR	32	0xC4	0xFC0B00C4	32-bit
ENET	PALR	32	0xE4	0xFC0B00E4	32-bit
ENET	PAUR	32	0xE8	0xFC0B00E8	32-bit
ENET	OPD	32	0xEC	0xFC0B00EC	32-bit
ENET	IAUR	32	0x118	0xFC0B0118	32-bit
ENET	IALR	32	0x11C	0xFC0B011C	32-bit
ENET	GAUR	32	0x120	0xFC0B0120	32-bit
ENET	GALR	32	0x124	0xFC0B0124	32-bit
ENET	TFWR	32	0x144	0xFC0B0144	32-bit
ENET	RDSR	32	0x180	0xFC0B0180	32-bit
ENET	TDSR	32	0x184	0xFC0B0184	32-bit
ENET	MRBR	32	0x188	0xFC0B0188	32-bit
ENET	RSFL	32	0x190	0xFC0B0190	32-bit
ENET	RSEM	32	0x194	0xFC0B0194	32-bit
ENET	RAEM	32	0x198	0xFC0B0198	32-bit
ENET	RAFL	32	0x19C	0xFC0B019C	32-bit
ENET	TSEM	32	0x1A0	0xFC0B01A0	32-bit
ENET	TAEM	32	0x1A4	0xFC0B01A4	32-bit
ENET	TAFL	32	0x1A8	0xFC0B01A8	32-bit
ENET	TIGP	32	0x1AC	0xFC0B01AC	32-bit
ENET	FTRL	32	0x1B0	0xFC0B01B0	32-bit
ENET	TACC	32	0x1C0	0xFC0B01C0	32-bit
ENET	RACC	32	0x1C4	0xFC0B01C4	32-bit
ENET	ATPER	32	0x40C	0xFC0B040C	32-bit
ENET	ATCOR	32	0x410	0xFC0B0410	32-bit
ENET	ATINC	32	0x414	0xFC0B0414	32-bit
eTIMER0	CH0_COMP1	16	0x0	0xFBC20000	16-bit
eTIMER0	CH0_COMP2	16	0x2	0xFBC20002	16-bit
eTIMER0	CH0_LOAD	16	0x8	0xFBC20008	16-bit
eTIMER0	CH0_CTRL1	16	0xE	0xFBC2000E	16-bit
eTIMER0	CH0_CTRL2	16	0x10	0xFBC20010	16-bit
eTIMER0	CH0_CTRL3	16	0x12	0xFBC20012	16-bit
eTIMER0	CH0_INTDMA	16	0x16	0xFBC20016	16-bit
eTIMER0	CH0_CMPLD1	16	0x18	0xFBC20018	16-bit
eTIMER0	CH0_CMPLD2	16	0x1A	0xFBC2001A	16-bit
eTIMER0	CH0_CCCTRL	16	0x1C	0xFBC2001C	16-bit
eTIMER0	CH0_FILT	16	0x1E	0xFBC2001E	16-bit
eTIMER0	CH1_COMP1	16	0x20	0xFBC20020	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
eTIMER0	CH1_COMP2	16	0x22	0xFBC20022	16-bit
eTIMER0	CH1_LOAD	16	0x28	0xFBC20028	16-bit
eTIMER0	CH1_CTRL1	16	0x2E	0xFBC2002E	16-bit
eTIMER0	CH1_CTRL2	16	0x30	0xFBC20030	16-bit
eTIMER0	CH1_CTRL3	16	0x32	0xFBC20032	16-bit
eTIMER0	CH1_INTDMA	16	0x36	0xFBC20036	16-bit
eTIMER0	CH1_CMPLD1	16	0x38	0xFBC20038	16-bit
eTIMER0	CH1_CMPLD2	16	0x3A	0xFBC2003A	16-bit
eTIMER0	CH1_CCCTRL	16	0x3C	0xFBC2003C	16-bit
eTIMER0	CH1_FILT	16	0x3E	0xFBC2003E	16-bit
eTIMER0	CH2_COMP1	16	0x40	0xFBC20040	16-bit
eTIMER0	CH2_COMP2	16	0x42	0xFBC20042	16-bit
eTIMER0	CH2_LOAD	16	0x48	0xFBC20048	16-bit
eTIMER0	CH2_CTRL1	16	0x4E	0xFBC2004E	16-bit
eTIMER0	CH2_CTRL2	16	0x50	0xFBC20050	16-bit
eTIMER0	CH2_CTRL3	16	0x52	0xFBC20052	16-bit
eTIMER0	CH2_INTDMA	16	0x56	0xFBC20056	16-bit
eTIMER0	CH2_CMPLD1	16	0x58	0xFBC20058	16-bit
eTIMER0	CH2_CMPLD2	16	0x5A	0xFBC2005A	16-bit
eTIMER0	CH2_CCCTRL	16	0x5C	0xFBC2005C	16-bit
eTIMER0	CH2_FILT	16	0x5E	0xFBC2005E	16-bit
eTIMER0	CH3_COMP1	16	0x60	0xFBC20060	16-bit
eTIMER0	CH3_COMP2	16	0x62	0xFBC20062	16-bit
eTIMER0	CH3_LOAD	16	0x68	0xFBC20068	16-bit
eTIMER0	CH3_CTRL1	16	0x6E	0xFBC2006E	16-bit
eTIMER0	CH3_CTRL2	16	0x70	0xFBC20070	16-bit
eTIMER0	CH3_CTRL3	16	0x72	0xFBC20072	16-bit
eTIMER0	CH3_INTDMA	16	0x76	0xFBC20076	16-bit
eTIMER0	CH3_CMPLD1	16	0x78	0xFBC20078	16-bit
eTIMER0	CH3_CMPLD2	16	0x7A	0xFBC2007A	16-bit
eTIMER0	CH3_CCCTRL	16	0x7C	0xFBC2007C	16-bit
eTIMER0	CH3_FILT	16	0x7E	0xFBC2007E	16-bit
eTIMER0	CH4_COMP1	16	0x80	0xFBC20080	16-bit
eTIMER0	CH4_COMP2	16	0x82	0xFBC20082	16-bit
eTIMER0	CH4_LOAD	16	0x88	0xFBC20088	16-bit
eTIMER0	CH4_CTRL1	16	0x8E	0xFBC2008E	16-bit
eTIMER0	CH4_CTRL2	16	0x90	0xFBC20090	16-bit
eTIMER0	CH4_CTRL3	16	0x92	0xFBC20092	16-bit
eTIMER0	CH4_INTDMA	16	0x96	0xFBC20096	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
eTIMER0	CH4_CMPLD1	16	0x98	0xFBC20098	16-bit
eTIMER0	CH4_CMPLD2	16	0x9A	0xFBC2009A	16-bit
eTIMER0	CH4_CCCTRL	16	0x9C	0xFBC2009C	16-bit
eTIMER0	CH4_FILT	16	0x9E	0xFBC2009E	16-bit
eTIMER0	CH5_COMP1	16	0xA0	0xFBC200A0	16-bit
eTIMER0	CH5_COMP2	16	0xA2	0xFBC200A2	16-bit
eTIMER0	CH5_LOAD	16	0xA8	0xFBC200A8	16-bit
eTIMER0	CH5_CTRL1	16	0xAE	0xFBC200AE	16-bit
eTIMER0	CH5_CTRL2	16	0xB0	0xFBC200B0	16-bit
eTIMER0	CH5_CTRL3	16	0xB2	0xFBC200B2	16-bit
eTIMER0	CH5_INTDMA	16	0xB6	0xFBC200B6	16-bit
eTIMER0	CH5_CMPLD1	16	0xB8	0xFBC200B8	16-bit
eTIMER0	CH5_CMPLD2	16	0xBA	0xFBC200BA	16-bit
eTIMER0	CH5_CCCTRL	16	0xBC	0xFBC200BC	16-bit
eTIMER0	CH5_FILT	16	0xBE	0xFBC200BE	16-bit
eTIMER0	WDTOL	16	0x100	0xFBC20100	16-bit
eTIMER0	WDTOH	16	0x102	0xFBC20102	16-bit
eTIMER0	ENBL	16	0x10C	0xFBC2010C	16-bit
eTIMER0	DREQ0	16	0x110	0xFBC20110	16-bit
eTIMER0	DREQ1	16	0x112	0xFBC20112	16-bit
eTIMER0	DREQ2	16	0x114	0xFBC20114	16-bit
eTIMER0	DREQ3	16	0x116	0xFBC20116	16-bit
eTIMER1	CH0_COMP1	16	0x0	0xFFC24000	16-bit
eTIMER1	CH0_COMP2	16	0x2	0xFFC24002	16-bit
eTIMER1	CH0_LOAD	16	0x8	0xFFC24008	16-bit
eTIMER1	CH0_CTRL1	16	0xE	0xFFC2400E	16-bit
eTIMER1	CH0_CTRL2	16	0x10	0xFFC24010	16-bit
eTIMER1	CH0_CTRL3	16	0x12	0xFFC24012	16-bit
eTIMER1	CH0_INTDMA	16	0x16	0xFFC24016	16-bit
eTIMER1	CH0_CMPLD1	16	0x18	0xFFC24018	16-bit
eTIMER1	CH0_CMPLD2	16	0x1A	0xFFC2401A	16-bit
eTIMER1	CH0_CCCTRL	16	0x1C	0xFFC2401C	16-bit
eTIMER1	CH0_FILT	16	0x1E	0xFFC2401E	16-bit
eTIMER1	CH1_COMP1	16	0x20	0xFFC24020	16-bit
eTIMER1	CH1_COMP2	16	0x22	0xFFC24022	16-bit
eTIMER1	CH1_LOAD	16	0x28	0xFFC24028	16-bit
eTIMER1	CH1_CTRL1	16	0x2E	0xFFC2402E	16-bit
eTIMER1	CH1_CTRL2	16	0x30	0xFFC24030	16-bit
eTIMER1	CH1_CTRL3	16	0x32	0xFFC24032	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
eTIMER1	CH1_INTDMA	16	0x36	0xFFC24036	16-bit
eTIMER1	CH1_CMPLD1	16	0x38	0xFFC24038	16-bit
eTIMER1	CH1_CMPLD2	16	0x3A	0xFFC2403A	16-bit
eTIMER1	CH1_CCCTRL	16	0x3C	0xFFC2403C	16-bit
eTIMER1	CH1_FILT	16	0x3E	0xFFC2403E	16-bit
eTIMER1	CH2_COMP1	16	0x40	0xFFC24040	16-bit
eTIMER1	CH2_COMP2	16	0x42	0xFFC24042	16-bit
eTIMER1	CH2_LOAD	16	0x48	0xFFC24048	16-bit
eTIMER1	CH2_CTRL1	16	0x4E	0xFFC2404E	16-bit
eTIMER1	CH2_CTRL2	16	0x50	0xFFC24050	16-bit
eTIMER1	CH2_CTRL3	16	0x52	0xFFC24052	16-bit
eTIMER1	CH2_INTDMA	16	0x56	0xFFC24056	16-bit
eTIMER1	CH2_CMPLD1	16	0x58	0xFFC24058	16-bit
eTIMER1	CH2_CMPLD2	16	0x5A	0xFFC2405A	16-bit
eTIMER1	CH2_CCCTRL	16	0x5C	0xFFC2405C	16-bit
eTIMER1	CH2_FILT	16	0x5E	0xFFC2405E	16-bit
eTIMER1	CH3_COMP1	16	0x60	0xFFC24060	16-bit
eTIMER1	CH3_COMP2	16	0x62	0xFFC24062	16-bit
eTIMER1	CH3_LOAD	16	0x68	0xFFC24068	16-bit
eTIMER1	CH3_CTRL1	16	0x6E	0xFFC2406E	16-bit
eTIMER1	CH3_CTRL2	16	0x70	0xFFC24070	16-bit
eTIMER1	CH3_CTRL3	16	0x72	0xFFC24072	16-bit
eTIMER1	CH3_INTDMA	16	0x76	0xFFC24076	16-bit
eTIMER1	CH3_CMPLD1	16	0x78	0xFFC24078	16-bit
eTIMER1	CH3_CMPLD2	16	0x7A	0xFFC2407A	16-bit
eTIMER1	CH3_CCCTRL	16	0x7C	0xFFC2407C	16-bit
eTIMER1	CH3_FILT	16	0x7E	0xFFC2407E	16-bit
eTIMER1	CH4_COMP1	16	0x80	0xFFC24080	16-bit
eTIMER1	CH4_COMP2	16	0x82	0xFFC24082	16-bit
eTIMER1	CH4_LOAD	16	0x88	0xFFC24088	16-bit
eTIMER1	CH4_CTRL1	16	0x8E	0xFFC2408E	16-bit
eTIMER1	CH4_CTRL2	16	0x90	0xFFC24090	16-bit
eTIMER1	CH4_CTRL3	16	0x92	0xFFC24092	16-bit
eTIMER1	CH4_INTDMA	16	0x96	0xFFC24096	16-bit
eTIMER1	CH4_CMPLD1	16	0x98	0xFFC24098	16-bit
eTIMER1	CH4_CMPLD2	16	0x9A	0xFFC2409A	16-bit
eTIMER1	CH4_CCCTRL	16	0x9C	0xFFC2409C	16-bit
eTIMER1	CH4_FILT	16	0x9E	0xFFC2409E	16-bit
eTIMER1	CH5_COMP1	16	0xA0	0xFFC240A0	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
eTIMER1	CH5_COMP2	16	0xA2	0xFFC240A2	16-bit
eTIMER1	CH5_LOAD	16	0xA8	0xFFC240A8	16-bit
eTIMER1	CH5_CTRL1	16	0xAE	0xFFC240AE	16-bit
eTIMER1	CH5_CTRL2	16	0xB0	0xFFC240B0	16-bit
eTIMER1	CH5_CTRL3	16	0xB2	0xFFC240B2	16-bit
eTIMER1	CH5_INTDMA	16	0xB6	0xFFC240B6	16-bit
eTIMER1	CH5_CMPLD1	16	0xB8	0xFFC240B8	16-bit
eTIMER1	CH5_CMPLD2	16	0xBA	0xFFC240BA	16-bit
eTIMER1	CH5_CCCTRL	16	0xBC	0xFFC240BC	16-bit
eTIMER1	CH5_FILT	16	0xBE	0xFFC240BE	16-bit
eTIMER1	ENBL	16	0x10C	0xFFC2410C	16-bit
eTIMER1	DREQ0	16	0x110	0xFFC24110	16-bit
eTIMER1	DREQ1	16	0x112	0xFFC24112	16-bit
eTIMER2	CH0_COMP1	16	0x0	0xFBC28000	16-bit
eTIMER2	CH0_COMP2	16	0x2	0xFBC28002	16-bit
eTIMER2	CH0_LOAD	16	0x8	0xFBC28008	16-bit
eTIMER2	CH0_CTRL1	16	0xE	0xFBC2800E	16-bit
eTIMER2	CH0_CTRL2	16	0x10	0xFBC28010	16-bit
eTIMER2	CH0_CTRL3	16	0x12	0xFBC28012	16-bit
eTIMER2	CH0_INTDMA	16	0x16	0xFBC28016	16-bit
eTIMER2	CH0_CMPLD1	16	0x18	0xFBC28018	16-bit
eTIMER2	CH0_CMPLD2	16	0x1A	0xFBC2801A	16-bit
eTIMER2	CH0_CCCTRL	16	0x1C	0xFBC2801C	16-bit
eTIMER2	CH0_FILT	16	0x1E	0xFBC2801E	16-bit
eTIMER2	CH1_COMP1	16	0x20	0xFBC28020	16-bit
eTIMER2	CH1_COMP2	16	0x22	0xFBC28022	16-bit
eTIMER2	CH1_LOAD	16	0x28	0xFBC28028	16-bit
eTIMER2	CH1_CTRL1	16	0x2E	0xFBC2802E	16-bit
eTIMER2	CH1_CTRL2	16	0x30	0xFBC28030	16-bit
eTIMER2	CH1_CTRL3	16	0x32	0xFBC28032	16-bit
eTIMER2	CH1_INTDMA	16	0x36	0xFBC28036	16-bit
eTIMER2	CH1_CMPLD1	16	0x38	0xFBC28038	16-bit
eTIMER2	CH1_CMPLD2	16	0x3A	0xFBC2803A	16-bit
eTIMER2	CH1_CCCTRL	16	0x3C	0xFBC2803C	16-bit
eTIMER2	CH1_FILT	16	0x3E	0xFBC2803E	16-bit
eTIMER2	CH2_COMP1	16	0x40	0xFBC28040	16-bit
eTIMER2	CH2_COMP2	16	0x42	0xFBC28042	16-bit
eTIMER2	CH2_LOAD	16	0x48	0xFBC28048	16-bit
eTIMER2	CH2_CTRL1	16	0x4E	0xFBC2804E	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
eTIMER2	CH2_CTRL2	16	0x50	0xFBC28050	16-bit
eTIMER2	CH2_CTRL3	16	0x52	0xFBC28052	16-bit
eTIMER2	CH2_INTDMA	16	0x56	0xFBC28056	16-bit
eTIMER2	CH2_CMPLD1	16	0x58	0xFBC28058	16-bit
eTIMER2	CH2_CMPLD2	16	0x5A	0xFBC2805A	16-bit
eTIMER2	CH2_CCCTRL	16	0x5C	0xFBC2805C	16-bit
eTIMER2	CH2_FILT	16	0x5E	0xFBC2805E	16-bit
eTIMER2	CH3_COMP1	16	0x60	0xFBC28060	16-bit
eTIMER2	CH3_COMP2	16	0x62	0xFBC28062	16-bit
eTIMER2	CH3_LOAD	16	0x68	0xFBC28068	16-bit
eTIMER2	CH3_CTRL1	16	0x6E	0xFBC2806E	16-bit
eTIMER2	CH3_CTRL2	16	0x70	0xFBC28070	16-bit
eTIMER2	CH3_CTRL3	16	0x72	0xFBC28072	16-bit
eTIMER2	CH3_INTDMA	16	0x76	0xFBC28076	16-bit
eTIMER2	CH3_CMPLD1	16	0x78	0xFBC28078	16-bit
eTIMER2	CH3_CMPLD2	16	0x7A	0xFBC2807A	16-bit
eTIMER2	CH3_CCCTRL	16	0x7C	0xFBC2807C	16-bit
eTIMER2	CH3_FILT	16	0x7E	0xFBC2807E	16-bit
eTIMER2	CH4_COMP1	16	0x80	0xFBC28080	16-bit
eTIMER2	CH4_COMP2	16	0x82	0xFBC28082	16-bit
eTIMER2	CH4_LOAD	16	0x88	0xFBC28088	16-bit
eTIMER2	CH4_CTRL1	16	0x8E	0xFBC2808E	16-bit
eTIMER2	CH4_CTRL2	16	0x90	0xFBC28090	16-bit
eTIMER2	CH4_CTRL3	16	0x92	0xFBC28092	16-bit
eTIMER2	CH4_INTDMA	16	0x96	0xFBC28096	16-bit
eTIMER2	CH4_CMPLD1	16	0x98	0xFBC28098	16-bit
eTIMER2	CH4_CMPLD2	16	0x9A	0xFBC2809A	16-bit
eTIMER2	CH4_CCCTRL	16	0x9C	0xFBC2809C	16-bit
eTIMER2	CH4_FILT	16	0x9E	0xFBC2809E	16-bit
eTIMER2	CH5_COMP1	16	0xA0	0xFBC280A0	16-bit
eTIMER2	CH5_COMP2	16	0xA2	0xFBC280A2	16-bit
eTIMER2	CH5_LOAD	16	0xA8	0xFBC280A8	16-bit
eTIMER2	CH5_CTRL1	16	0xAE	0xFBC280AE	16-bit
eTIMER2	CH5_CTRL2	16	0xB0	0xFBC280B0	16-bit
eTIMER2	CH5_CTRL3	16	0xB2	0xFBC280B2	16-bit
eTIMER2	CH5_INTDMA	16	0xB6	0xFBC280B6	16-bit
eTIMER2	CH5_CMPLD1	16	0xB8	0xFBC280B8	16-bit
eTIMER2	CH5_CMPLD2	16	0xBA	0xFBC280BA	16-bit
eTIMER2	CH5_CCCTRL	16	0xBC	0xFBC280BC	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
eTIMER2	CH5_FILT	16	0xBE	0xFBC280BE	16-bit
eTIMER2	ENBL	16	0x10C	0xFBC2810C	16-bit
eTIMER2	DREQ0	16	0x110	0xFBC28110	16-bit
eTIMER2	DREQ1	16	0x112	0xFBC28112	16-bit
FCCU	CFG	32	0x8	0xFBF58008	32-bit
FCCU	NCF_CFG0	32	0x1C	0xFBF5801C	32-bit
FCCU	NCF_CFG1	32	0x20	0xFBF58020	32-bit
FCCU	NCFS_CFG0	32	0x4C	0xFBF5804C	32-bit
FCCU	NCFS_CFG1	32	0x50	0xFBF58050	32-bit
FCCU	NCF_E0	32	0x94	0xFBF58094	32-bit
FCCU	NCF_E1	32	0x98	0xFBF58098	32-bit
FCCU	NCF_TOE0	32	0xA4	0xFBF580A4	32-bit
FCCU	NCF_TOE1	32	0xA8	0xFBF580A8	32-bit
FCCU	NCF_TO	32	0xB4	0xFBF580B4	32-bit
FCCU	CFG_TO	32	0xB8	0xFBF580B8	32-bit
FCCU	NCFF	32	0xDC	0xFBF580DC	32-bit
FCCU	IRQ_EN	32	0xE4	0xFBF580E4	32-bit
FCCU	TRANS_LOCK	32	0xF0	0xFBF580F0	32-bit
FCCU	PERMNT_LOCK	32	0xF4	0xFBF580F4	32-bit
C55FMC	MCR	32	0x0	0xFFFE0000	32-bit
C55FMC	LOCK0	32	0x10	0xFFFE0010	32-bit
C55FMC	LOCK1	32	0x14	0xFFFE0014	32-bit
C55FMC	LOCK2	32	0x18	0xFFFE0018	32-bit
C55FMC	UT0	32	0x54	0xFFFE0054	32-bit
C55FMC	UM0	32	0x58	0xFFFE0058	32-bit
C55FMC	UM1	32	0x5C	0xFFFE005C	32-bit
C55FMC	UM2	32	0x60	0xFFFE0060	32-bit
C55FMC	UM3	32	0x64	0xFFFE0064	32-bit
C55FMC	UM4	32	0x68	0xFFFE0068	32-bit
C55FMC	UM5	32	0x6C	0xFFFE006C	32-bit
C55FMC	UM6	32	0x70	0xFFFE0070	32-bit
C55FMC	UM7	32	0x74	0xFFFE0074	32-bit
C55FMC	UM8	32	0x78	0xFFFE0078	32-bit
C55FMC	UM9	32	0x7C	0xFFFE007C	32-bit
FlexCAN_0	MCR	32	0x0	0xFFEC0000	32-bit
FlexCAN_0	CTRL1	32	0x4	0xFFEC0004	32-bit
FlexCAN_0	RXMGMASK	32	0x10	0xFFEC0010	32-bit
FlexCAN_0	RX14MASK	32	0x14	0xFFEC0014	32-bit
FlexCAN_0	RX15MASK	32	0x18	0xFFEC0018	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_0	IMASK2	32	0x24	0xFFEC0024	32-bit
FlexCAN_0	IMASK1	32	0x28	0xFFEC0028	32-bit
FlexCAN_0	CTRL2	32	0x34	0xFFEC0034	32-bit
FlexCAN_0	MSG0_CS	32	0x80	0xFFEC0080	32-bit
FlexCAN_0	MSG0_ID	32	0x84	0xFFEC0084	32-bit
FlexCAN_0	MSG1_CS	32	0x90	0xFFEC0090	32-bit
FlexCAN_0	MSG1_ID	32	0x94	0xFFEC0094	32-bit
FlexCAN_0	MSG2_CS	32	0x00A0	0xFFEC00A0	32-bit
FlexCAN_0	MSG2_ID	32	0x00A4	0xFFEC00A4	32-bit
FlexCAN_0	MSG3_CS	32	0x00B0	0xFFEC00B0	32-bit
FlexCAN_0	MSG3_ID	32	0x00B4	0xFFEC00B4	32-bit
FlexCAN_0	MSG4_CS	32	0x00C0	0xFFEC00C0	32-bit
FlexCAN_0	MSG4_ID	32	0x00C4	0xFFEC00C4	32-bit
FlexCAN_0	MSG5_CS	32	0x00D0	0xFFEC00D0	32-bit
FlexCAN_0	MSG5_ID	32	0x00D4	0xFFEC00D4	32-bit
FlexCAN_0	MSG6_CS	32	0x00E0	0xFFEC00E0	32-bit
FlexCAN_0	MSG6_ID	32	0x00E4	0xFFEC00E4	32-bit
FlexCAN_0	MSG7_CS	32	0x00F0	0xFFEC00F0	32-bit
FlexCAN_0	MSG7_ID	32	0x00F4	0xFFEC00F4	32-bit
FlexCAN_0	MSG8_CS	32	0x100	0xFFEC0100	32-bit
FlexCAN_0	MSG8_ID	32	0x104	0xFFEC0104	32-bit
FlexCAN_0	MSG9_CS	32	0x110	0xFFEC0110	32-bit
FlexCAN_0	MSG9_ID	32	0x114	0xFFEC0114	32-bit
FlexCAN_0	MSG10_CS	32	0x120	0xFFEC0120	32-bit
FlexCAN_0	MSG10_ID	32	0x124	0xFFEC0124	32-bit
FlexCAN_0	MSG11_CS	32	0x130	0xFFEC0130	32-bit
FlexCAN_0	MSG11_ID	32	0x134	0xFFEC0134	32-bit
FlexCAN_0	MSG12_CS	32	0x140	0xFFEC0140	32-bit
FlexCAN_0	MSG12_ID	32	0x144	0xFFEC0144	32-bit
FlexCAN_0	MSG13_CS	32	0x150	0xFFEC0150	32-bit
FlexCAN_0	MSG13_ID	32	0x154	0xFFEC0154	32-bit
FlexCAN_0	MSG14_CS	32	0x160	0xFFEC0160	32-bit
FlexCAN_0	MSG14_ID	32	0x164	0xFFEC0164	32-bit
FlexCAN_0	MSG15_CS	32	0x170	0xFFEC0170	32-bit
FlexCAN_0	MSG15_ID	32	0x174	0xFFEC0174	32-bit
FlexCAN_0	MSG16_CS	32	0x180	0xFFEC0180	32-bit
FlexCAN_0	MSG16_ID	32	0x184	0xFFEC0184	32-bit
FlexCAN_0	MSG17_CS	32	0x190	0xFFEC0190	32-bit
FlexCAN_0	MSG17_ID	32	0x194	0xFFEC0194	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_0	MSG18_CS	32	0x01A0	0xFFEC01A0	32-bit
FlexCAN_0	MSG18_ID	32	0x01A4	0xFFEC01A4	32-bit
FlexCAN_0	MSG19_CS	32	0x01B0	0xFFEC01B0	32-bit
FlexCAN_0	MSG19_ID	32	0x01B4	0xFFEC01B4	32-bit
FlexCAN_0	MSG20_CS	32	0x01C0	0xFFEC01C0	32-bit
FlexCAN_0	MSG20_ID	32	0x01C4	0xFFEC01C4	32-bit
FlexCAN_0	MSG21_CS	32	0x01D0	0xFFEC01D0	32-bit
FlexCAN_0	MSG21_ID	32	0x01D4	0xFFEC01D4	32-bit
FlexCAN_0	MSG22_CS	32	0x01E0	0xFFEC01E0	32-bit
FlexCAN_0	MSG22_ID	32	0x01E4	0xFFEC01E4	32-bit
FlexCAN_0	MSG23_CS	32	0x01F0	0xFFEC01F0	32-bit
FlexCAN_0	MSG23_ID	32	0x01F4	0xFFEC01F4	32-bit
FlexCAN_0	MSG24_CS	32	0x200	0xFFEC0200	32-bit
FlexCAN_0	MSG24_ID	32	0x204	0xFFEC0204	32-bit
FlexCAN_0	MSG25_CS	32	0x210	0xFFEC0210	32-bit
FlexCAN_0	MSG25_ID	32	0x214	0xFFEC0214	32-bit
FlexCAN_0	MSG26_CS	32	0x220	0xFFEC0220	32-bit
FlexCAN_0	MSG26_ID	32	0x224	0xFFEC0224	32-bit
FlexCAN_0	MSG27_CS	32	0x230	0xFFEC0230	32-bit
FlexCAN_0	MSG27_ID	32	0x234	0xFFEC0234	32-bit
FlexCAN_0	MSG28_CS	32	0x240	0xFFEC0240	32-bit
FlexCAN_0	MSG28_ID	32	0x244	0xFFEC0244	32-bit
FlexCAN_0	MSG29_CS	32	0x250	0xFFEC0250	32-bit
FlexCAN_0	MSG29_ID	32	0x254	0xFFEC0254	32-bit
FlexCAN_0	MSG30_CS	32	0x260	0xFFEC0260	32-bit
FlexCAN_0	MSG30_ID	32	0x264	0xFFEC0264	32-bit
FlexCAN_0	MSG31_CS	32	0x270	0xFFEC0270	32-bit
FlexCAN_0	MSG31_ID	32	0x274	0xFFEC0274	32-bit
FlexCAN_0	MSG32_CS	32	0x280	0xFFEC0280	32-bit
FlexCAN_0	MSG32_ID	32	0x284	0xFFEC0284	32-bit
FlexCAN_0	MSG33_CS	32	0x290	0xFFEC0290	32-bit
FlexCAN_0	MSG33_ID	32	0x294	0xFFEC0294	32-bit
FlexCAN_0	MSG34_CS	32	0x2a0	0xFFEC02A0	32-bit
FlexCAN_0	MSG34_ID	32	0x2a4	0xFFEC02A4	32-bit
FlexCAN_0	MSG35_CS	32	0x2b0	0xFFEC02B0	32-bit
FlexCAN_0	MSG35_ID	32	0x2b4	0xFFEC02B4	32-bit
FlexCAN_0	MSG36_CS	32	0x2c0	0xFFEC02C0	32-bit
FlexCAN_0	MSG36_ID	32	0x2c4	0xFFEC02C4	32-bit
FlexCAN_0	MSG37_CS	32	0x2d0	0xFFEC02D0	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_0	MSG37_ID	32	0x2d4	0xFFEC02D4	32-bit
FlexCAN_0	MSG38_CS	32	0x2e0	0xFFEC02E0	32-bit
FlexCAN_0	MSG38_ID	32	0x2e4	0xFFEC02E4	32-bit
FlexCAN_0	MSG39_CS	32	0x2f0	0xFFEC02F0	32-bit
FlexCAN_0	MSG39_ID	32	0x2f4	0xFFEC02F4	32-bit
FlexCAN_0	MSG40_CS	32	0x300	0xFFEC0300	32-bit
FlexCAN_0	MSG40_ID	32	0x304	0xFFEC0304	32-bit
FlexCAN_0	MSG41_CS	32	0x310	0xFFEC0310	32-bit
FlexCAN_0	MSG41_ID	32	0x314	0xFFEC0314	32-bit
FlexCAN_0	MSG42_CS	32	0x320	0xFFEC0320	32-bit
FlexCAN_0	MSG42_ID	32	0x324	0xFFEC0324	32-bit
FlexCAN_0	MSG43_CS	32	0x330	0xFFEC0330	32-bit
FlexCAN_0	MSG43_ID	32	0x334	0xFFEC0334	32-bit
FlexCAN_0	MSG44_CS	32	0x340	0xFFEC0340	32-bit
FlexCAN_0	MSG44_ID	32	0x344	0xFFEC0344	32-bit
FlexCAN_0	MSG45_CS	32	0x350	0xFFEC0350	32-bit
FlexCAN_0	MSG45_ID	32	0x354	0xFFEC0354	32-bit
FlexCAN_0	MSG46_CS	32	0x360	0xFFEC0360	32-bit
FlexCAN_0	MSG46_ID	32	0x364	0xFFEC0364	32-bit
FlexCAN_0	MSG47_CS	32	0x370	0xFFEC0370	32-bit
FlexCAN_0	MSG47_ID	32	0x374	0xFFEC0374	32-bit
FlexCAN_0	MSG48_CS	32	0x380	0xFFEC0380	32-bit
FlexCAN_0	MSG48_ID	32	0x384	0xFFEC0384	32-bit
FlexCAN_0	MSG49_CS	32	0x390	0xFFEC0390	32-bit
FlexCAN_0	MSG49_ID	32	0x394	0xFFEC0394	32-bit
FlexCAN_0	MSG50_CS	32	0x3a0	0xFFEC03A0	32-bit
FlexCAN_0	MSG50_ID	32	0x3a4	0xFFEC03A4	32-bit
FlexCAN_0	MSG51_CS	32	0x3b0	0xFFEC03B0	32-bit
FlexCAN_0	MSG51_ID	32	0x3b4	0xFFEC03B4	32-bit
FlexCAN_0	MSG52_CS	32	0x3c0	0xFFEC03C0	32-bit
FlexCAN_0	MSG52_ID	32	0x3c4	0xFFEC03C4	32-bit
FlexCAN_0	MSG53_CS	32	0x3d0	0xFFEC03D0	32-bit
FlexCAN_0	MSG53_ID	32	0x3d4	0xFFEC03D4	32-bit
FlexCAN_0	MSG54_CS	32	0x3e0	0xFFEC03E0	32-bit
FlexCAN_0	MSG54_ID	32	0x3e4	0xFFEC03E4	32-bit
FlexCAN_0	MSG55_CS	32	0x3f0	0xFFEC03F0	32-bit
FlexCAN_0	MSG55_ID	32	0x3f4	0xFFEC03F4	32-bit
FlexCAN_0	MSG56_CS	32	0x400	0xFFEC0400	32-bit
FlexCAN_0	MSG56_ID	32	0x404	0xFFEC0404	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_0	MSG57_CS	32	0x410	0xFFEC0410	32-bit
FlexCAN_0	MSG57_ID	32	0x414	0xFFEC0414	32-bit
FlexCAN_0	MSG58_CS	32	0x420	0xFFEC0420	32-bit
FlexCAN_0	MSG58_ID	32	0x424	0xFFEC0424	32-bit
FlexCAN_0	MSG59_CS	32	0x430	0xFFEC0430	32-bit
FlexCAN_0	MSG59_ID	32	0x434	0xFFEC0434	32-bit
FlexCAN_0	MSG60_CS	32	0x440	0xFFEC0440	32-bit
FlexCAN_0	MSG60_ID	32	0x444	0xFFEC0444	32-bit
FlexCAN_0	MSG61_CS	32	0x450	0xFFEC0450	32-bit
FlexCAN_0	MSG61_ID	32	0x454	0xFFEC0454	32-bit
FlexCAN_0	MSG62_CS	32	0x460	0xFFEC0460	32-bit
FlexCAN_0	MSG62_ID	32	0x464	0xFFEC0464	32-bit
FlexCAN_0	MSG63_CS	32	0x470	0xFFEC0470	32-bit
FlexCAN_0	MSG63_ID	32	0x474	0xFFEC0474	32-bit
FlexCAN_0	RXIMR0	32	0x880	0xFFEC0880	32-bit
FlexCAN_0	RXIMR1	32	0x884	0xFFEC0884	32-bit
FlexCAN_0	RXIMR2	32	0x888	0xFFEC0888	32-bit
FlexCAN_0	RXIMR3	32	0x088C	0xFFEC088C	32-bit
FlexCAN_0	RXIMR4	32	0x890	0xFFEC0890	32-bit
FlexCAN_0	RXIMR5	32	0x894	0xFFEC0894	32-bit
FlexCAN_0	RXIMR6	32	0x898	0xFFEC0898	32-bit
FlexCAN_0	RXIMR7	32	0x089C	0xFFEC089C	32-bit
FlexCAN_0	RXIMR8	32	0x08A0	0xFFEC08A0	32-bit
FlexCAN_0	RXIMR9	32	0x08A4	0xFFEC08A4	32-bit
FlexCAN_0	RXIMR10	32	0x08A8	0xFFEC08A8	32-bit
FlexCAN_0	RXIMR11	32	0x08AC	0xFFEC08AC	32-bit
FlexCAN_0	RXIMR12	32	0x08B0	0xFFEC08B0	32-bit
FlexCAN_0	RXIMR13	32	0x08B4	0xFFEC08B4	32-bit
FlexCAN_0	RXIMR14	32	0x08B8	0xFFEC08B8	32-bit
FlexCAN_0	RXIMR15	32	0x08BC	0xFFEC08BC	32-bit
FlexCAN_0	RXIMR16	32	0x08C0	0xFFEC08C0	32-bit
FlexCAN_0	RXIMR17	32	0x08C4	0xFFEC08C4	32-bit
FlexCAN_0	RXIMR18	32	0x08C8	0xFFEC08C8	32-bit
FlexCAN_0	RXIMR19	32	0x08CC	0xFFEC08CC	32-bit
FlexCAN_0	RXIMR20	32	0x08D0	0xFFEC08D0	32-bit
FlexCAN_0	RXIMR21	32	0x08D4	0xFFEC08D4	32-bit
FlexCAN_0	RXIMR22	32	0x08D8	0xFFEC08D8	32-bit
FlexCAN_0	RXIMR23	32	0x08DC	0xFFEC08DC	32-bit
FlexCAN_0	RXIMR24	32	0x8e0	0xFFEC08E0	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_0	RXIMR25	32	0x8e4	0xFFEC08E4	32-bit
FlexCAN_0	RXIMR26	32	0x8e8	0xFFEC08E8	32-bit
FlexCAN_0	RXIMR27	32	0x08EC	0xFFEC08EC	32-bit
FlexCAN_0	RXIMR28	32	0x08F0	0xFFEC08F0	32-bit
FlexCAN_0	RXIMR29	32	0x08F4	0xFFEC08F4	32-bit
FlexCAN_0	RXIMR30	32	0x08F8	0xFFEC08F8	32-bit
FlexCAN_0	RXIMR31	32	0x08FC	0xFFEC08FC	32-bit
FlexCAN_0	FCMECR	32	0x0AE0	0xFFEC0AE0	32-bit
FlexCAN_1	MCR	32	0x0	0xFFEC4000	32-bit
FlexCAN_1	CTRL1	32	0x4	0xFFEC4004	32-bit
FlexCAN_1	RXMGMASK	32	0x10	0xFFEC4010	32-bit
FlexCAN_1	RX14MASK	32	0x14	0xFFEC4014	32-bit
FlexCAN_1	RX15MASK	32	0x18	0xFFEC4018	32-bit
FlexCAN_1	IMASK2	32	0x24	0xFFEC4024	32-bit
FlexCAN_1	IMASK1	32	0x28	0xFFEC4028	32-bit
FlexCAN_1	CTRL2	32	0x34	0xFFEC4034	32-bit
FlexCAN_1	MSG0_CS	32	0x80	0xFFEC4080	32-bit
FlexCAN_1	MSG0_ID	32	0x84	0xFFEC4084	32-bit
FlexCAN_1	MSG1_CS	32	0x90	0xFFEC4090	32-bit
FlexCAN_1	MSG1_ID	32	0x94	0xFFEC4094	32-bit
FlexCAN_1	MSG2_CS	32	0x00A0	0xFFEC40A0	32-bit
FlexCAN_1	MSG2_ID	32	0x00A4	0xFFEC40A4	32-bit
FlexCAN_1	MSG3_CS	32	0x00B0	0xFFEC40B0	32-bit
FlexCAN_1	MSG3_ID	32	0x00B4	0xFFEC40B4	32-bit
FlexCAN_1	MSG4_CS	32	0x00C0	0xFFEC40C0	32-bit
FlexCAN_1	MSG4_ID	32	0x00C4	0xFFEC40C4	32-bit
FlexCAN_1	MSG5_CS	32	0x00D0	0xFFEC40D0	32-bit
FlexCAN_1	MSG5_ID	32	0x00D4	0xFFEC40D4	32-bit
FlexCAN_1	MSG6_CS	32	0x00E0	0xFFEC40E0	32-bit
FlexCAN_1	MSG6_ID	32	0x00E4	0xFFEC40E4	32-bit
FlexCAN_1	MSG7_CS	32	0x00F0	0xFFEC40F0	32-bit
FlexCAN_1	MSG7_ID	32	0x00F4	0xFFEC40F4	32-bit
FlexCAN_1	MSG8_CS	32	0x100	0xFFEC4100	32-bit
FlexCAN_1	MSG8_ID	32	0x104	0xFFEC4104	32-bit
FlexCAN_1	MSG9_CS	32	0x110	0xFFEC4110	32-bit
FlexCAN_1	MSG9_ID	32	0x114	0xFFEC4114	32-bit
FlexCAN_1	MSG10_CS	32	0x120	0xFFEC4120	32-bit
FlexCAN_1	MSG10_ID	32	0x124	0xFFEC4124	32-bit
FlexCAN_1	MSG11_CS	32	0x130	0xFFEC4130	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_1	MSG11_ID	32	0x134	0xFFEC4134	32-bit
FlexCAN_1	MSG12_CS	32	0x140	0xFFEC4140	32-bit
FlexCAN_1	MSG12_ID	32	0x144	0xFFEC4144	32-bit
FlexCAN_1	MSG13_CS	32	0x150	0xFFEC4150	32-bit
FlexCAN_1	MSG13_ID	32	0x154	0xFFEC4154	32-bit
FlexCAN_1	MSG14_CS	32	0x160	0xFFEC4160	32-bit
FlexCAN_1	MSG14_ID	32	0x164	0xFFEC4164	32-bit
FlexCAN_1	MSG15_CS	32	0x170	0xFFEC4170	32-bit
FlexCAN_1	MSG15_ID	32	0x174	0xFFEC4174	32-bit
FlexCAN_1	MSG16_CS	32	0x180	0xFFEC4180	32-bit
FlexCAN_1	MSG16_ID	32	0x184	0xFFEC4184	32-bit
FlexCAN_1	MSG17_CS	32	0x190	0xFFEC4190	32-bit
FlexCAN_1	MSG17_ID	32	0x194	0xFFEC4194	32-bit
FlexCAN_1	MSG18_CS	32	0x01A0	0xFFEC41A0	32-bit
FlexCAN_1	MSG18_ID	32	0x01A4	0xFFEC41A4	32-bit
FlexCAN_1	MSG19_CS	32	0x01B0	0xFFEC41B0	32-bit
FlexCAN_1	MSG19_ID	32	0x01B4	0xFFEC41B4	32-bit
FlexCAN_1	MSG20_CS	32	0x01C0	0xFFEC41C0	32-bit
FlexCAN_1	MSG20_ID	32	0x01C4	0xFFEC41C4	32-bit
FlexCAN_1	MSG21_CS	32	0x01D0	0xFFEC41D0	32-bit
FlexCAN_1	MSG21_ID	32	0x01D4	0xFFEC41D4	32-bit
FlexCAN_1	MSG22_CS	32	0x01E0	0xFFEC41E0	32-bit
FlexCAN_1	MSG22_ID	32	0x01E4	0xFFEC41E4	32-bit
FlexCAN_1	MSG23_CS	32	0x01F0	0xFFEC41F0	32-bit
FlexCAN_1	MSG23_ID	32	0x01F4	0xFFEC41F4	32-bit
FlexCAN_1	MSG24_CS	32	0x200	0xFFEC4200	32-bit
FlexCAN_1	MSG24_ID	32	0x204	0xFFEC4204	32-bit
FlexCAN_1	MSG25_CS	32	0x210	0xFFEC4210	32-bit
FlexCAN_1	MSG25_ID	32	0x214	0xFFEC4214	32-bit
FlexCAN_1	MSG26_CS	32	0x220	0xFFEC4220	32-bit
FlexCAN_1	MSG26_ID	32	0x224	0xFFEC4224	32-bit
FlexCAN_1	MSG27_CS	32	0x230	0xFFEC4230	32-bit
FlexCAN_1	MSG27_ID	32	0x234	0xFFEC4234	32-bit
FlexCAN_1	MSG28_CS	32	0x240	0xFFEC4240	32-bit
FlexCAN_1	MSG28_ID	32	0x244	0xFFEC4244	32-bit
FlexCAN_1	MSG29_CS	32	0x250	0xFFEC4250	32-bit
FlexCAN_1	MSG29_ID	32	0x254	0xFFEC4254	32-bit
FlexCAN_1	MSG30_CS	32	0x260	0xFFEC4260	32-bit
FlexCAN_1	MSG30_ID	32	0x264	0xFFEC4264	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_1	MSG31_CS	32	0x270	0xFFEC4270	32-bit
FlexCAN_1	MSG31_ID	32	0x274	0xFFEC4274	32-bit
FlexCAN_1	MSG32_CS	32	0x280	0xFFEC4280	32-bit
FlexCAN_1	MSG32_ID	32	0x284	0xFFEC4284	32-bit
FlexCAN_1	MSG33_CS	32	0x290	0xFFEC4290	32-bit
FlexCAN_1	MSG33_ID	32	0x294	0xFFEC4294	32-bit
FlexCAN_1	MSG34_CS	32	0x2a0	0xFFEC42A0	32-bit
FlexCAN_1	MSG34_ID	32	0x2a4	0xFFEC42A4	32-bit
FlexCAN_1	MSG35_CS	32	0x2b0	0xFFEC42B0	32-bit
FlexCAN_1	MSG35_ID	32	0x2b4	0xFFEC42B4	32-bit
FlexCAN_1	MSG36_CS	32	0x2c0	0xFFEC42C0	32-bit
FlexCAN_1	MSG36_ID	32	0x2c4	0xFFEC42C4	32-bit
FlexCAN_1	MSG37_CS	32	0x2d0	0xFFEC42D0	32-bit
FlexCAN_1	MSG37_ID	32	0x2d4	0xFFEC42D4	32-bit
FlexCAN_1	MSG38_CS	32	0x2e0	0xFFEC42E0	32-bit
FlexCAN_1	MSG38_ID	32	0x2e4	0xFFEC42E4	32-bit
FlexCAN_1	MSG39_CS	32	0x2f0	0xFFEC42F0	32-bit
FlexCAN_1	MSG39_ID	32	0x2f4	0xFFEC42F4	32-bit
FlexCAN_1	MSG40_CS	32	0x300	0xFFEC4300	32-bit
FlexCAN_1	MSG40_ID	32	0x304	0xFFEC4304	32-bit
FlexCAN_1	MSG41_CS	32	0x310	0xFFEC4310	32-bit
FlexCAN_1	MSG41_ID	32	0x314	0xFFEC4314	32-bit
FlexCAN_1	MSG42_CS	32	0x320	0xFFEC4320	32-bit
FlexCAN_1	MSG42_ID	32	0x324	0xFFEC4324	32-bit
FlexCAN_1	MSG43_CS	32	0x330	0xFFEC4330	32-bit
FlexCAN_1	MSG43_ID	32	0x334	0xFFEC4334	32-bit
FlexCAN_1	MSG44_CS	32	0x340	0xFFEC4340	32-bit
FlexCAN_1	MSG44_ID	32	0x344	0xFFEC4344	32-bit
FlexCAN_1	MSG45_CS	32	0x350	0xFFEC4350	32-bit
FlexCAN_1	MSG45_ID	32	0x354	0xFFEC4354	32-bit
FlexCAN_1	MSG46_CS	32	0x360	0xFFEC4360	32-bit
FlexCAN_1	MSG46_ID	32	0x364	0xFFEC4364	32-bit
FlexCAN_1	MSG47_CS	32	0x370	0xFFEC4370	32-bit
FlexCAN_1	MSG47_ID	32	0x374	0xFFEC4374	32-bit
FlexCAN_1	MSG48_CS	32	0x380	0xFFEC4380	32-bit
FlexCAN_1	MSG48_ID	32	0x384	0xFFEC4384	32-bit
FlexCAN_1	MSG49_CS	32	0x390	0xFFEC4390	32-bit
FlexCAN_1	MSG49_ID	32	0x394	0xFFEC4394	32-bit
FlexCAN_1	MSG50_CS	32	0x3a0	0xFFEC43A0	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_1	MSG50_ID	32	0x3a4	0xFFEC43A4	32-bit
FlexCAN_1	MSG51_CS	32	0x3b0	0xFFEC43B0	32-bit
FlexCAN_1	MSG51_ID	32	0x3b4	0xFFEC43B4	32-bit
FlexCAN_1	MSG52_CS	32	0x3c0	0xFFEC43C0	32-bit
FlexCAN_1	MSG52_ID	32	0x3c4	0xFFEC43C4	32-bit
FlexCAN_1	MSG53_CS	32	0x3d0	0xFFEC43D0	32-bit
FlexCAN_1	MSG53_ID	32	0x3d4	0xFFEC43D4	32-bit
FlexCAN_1	MSG54_CS	32	0x3e0	0xFFEC43E0	32-bit
FlexCAN_1	MSG54_ID	32	0x3e4	0xFFEC43E4	32-bit
FlexCAN_1	MSG55_CS	32	0x3f0	0xFFEC43F0	32-bit
FlexCAN_1	MSG55_ID	32	0x3f4	0xFFEC43F4	32-bit
FlexCAN_1	MSG56_CS	32	0x400	0xFFEC4400	32-bit
FlexCAN_1	MSG56_ID	32	0x404	0xFFEC4404	32-bit
FlexCAN_1	MSG57_CS	32	0x410	0xFFEC4410	32-bit
FlexCAN_1	MSG57_ID	32	0x414	0xFFEC4414	32-bit
FlexCAN_1	MSG58_CS	32	0x420	0xFFEC4420	32-bit
FlexCAN_1	MSG58_ID	32	0x424	0xFFEC4424	32-bit
FlexCAN_1	MSG59_CS	32	0x430	0xFFEC4430	32-bit
FlexCAN_1	MSG59_ID	32	0x434	0xFFEC4434	32-bit
FlexCAN_1	MSG60_CS	32	0x440	0xFFEC4440	32-bit
FlexCAN_1	MSG60_ID	32	0x444	0xFFEC4444	32-bit
FlexCAN_1	MSG61_CS	32	0x450	0xFFEC4450	32-bit
FlexCAN_1	MSG61_ID	32	0x454	0xFFEC4454	32-bit
FlexCAN_1	MSG62_CS	32	0x460	0xFFEC4460	32-bit
FlexCAN_1	MSG62_ID	32	0x464	0xFFEC4464	32-bit
FlexCAN_1	MSG63_CS	32	0x470	0xFFEC4470	32-bit
FlexCAN_1	MSG63_ID	32	0x474	0xFFEC4474	32-bit
FlexCAN_1	RXIMR0	32	0x880	0xFFEC4880	32-bit
FlexCAN_1	RXIMR1	32	0x884	0xFFEC4884	32-bit
FlexCAN_1	RXIMR2	32	0x888	0xFFEC4888	32-bit
FlexCAN_1	RXIMR3	32	0x088C	0xFFEC488C	32-bit
FlexCAN_1	RXIMR4	32	0x890	0xFFEC4890	32-bit
FlexCAN_1	RXIMR5	32	0x894	0xFFEC4894	32-bit
FlexCAN_1	RXIMR6	32	0x898	0xFFEC4898	32-bit
FlexCAN_1	RXIMR7	32	0x089C	0xFFEC489C	32-bit
FlexCAN_1	RXIMR8	32	0x08A0	0xFFEC48A0	32-bit
FlexCAN_1	RXIMR9	32	0x08A4	0xFFEC48A4	32-bit
FlexCAN_1	RXIMR10	32	0x08A8	0xFFEC48A8	32-bit
FlexCAN_1	RXIMR11	32	0x08AC	0xFFEC48AC	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_1	RXIMR12	32	0x08B0	0xFFEC48B0	32-bit
FlexCAN_1	RXIMR13	32	0x08B4	0xFFEC48B4	32-bit
FlexCAN_1	RXIMR14	32	0x08B8	0xFFEC48B8	32-bit
FlexCAN_1	RXIMR15	32	0x08BC	0xFFEC48BC	32-bit
FlexCAN_1	RXIMR16	32	0x08C0	0xFFEC48C0	32-bit
FlexCAN_1	RXIMR17	32	0x08C4	0xFFEC48C4	32-bit
FlexCAN_1	RXIMR18	32	0x08C8	0xFFEC48C8	32-bit
FlexCAN_1	RXIMR19	32	0x08CC	0xFFEC48CC	32-bit
FlexCAN_1	RXIMR20	32	0x08D0	0xFFEC48D0	32-bit
FlexCAN_1	RXIMR21	32	0x08D4	0xFFEC48D4	32-bit
FlexCAN_1	RXIMR22	32	0x08D8	0xFFEC48D8	32-bit
FlexCAN_1	RXIMR23	32	0x08DC	0xFFEC48DC	32-bit
FlexCAN_1	RXIMR24	32	0x8e0	0xFFEC48E0	32-bit
FlexCAN_1	RXIMR25	32	0x8e4	0xFFEC48E4	32-bit
FlexCAN_1	RXIMR26	32	0x8e8	0xFFEC48E8	32-bit
FlexCAN_1	RXIMR27	32	0x08EC	0xFFEC48EC	32-bit
FlexCAN_1	RXIMR28	32	0x08F0	0xFFEC48F0	32-bit
FlexCAN_1	RXIMR29	32	0x08F4	0xFFEC48F4	32-bit
FlexCAN_1	RXIMR30	32	0x08F8	0xFFEC48F8	32-bit
FlexCAN_1	RXIMR31	32	0x08FC	0xFFEC48FC	32-bit
FlexCAN_1	FCMECR	32	0x0AE0	0xFFEC4AE0	32-bit
FlexCAN_2	MCR	32	0x0	0xFFEC8000	32-bit
FlexCAN_2	CTRL1	32	0x4	0xFFEC8004	32-bit
FlexCAN_2	RXMGMASK	32	0x10	0xFFEC8010	32-bit
FlexCAN_2	RX14MASK	32	0x14	0xFFEC8014	32-bit
FlexCAN_2	RX15MASK	32	0x18	0xFFEC8018	32-bit
FlexCAN_2	IMASK2	32	0x24	0xFFEC8024	32-bit
FlexCAN_2	IMASK1	32	0x28	0xFFEC8028	32-bit
FlexCAN_2	CTRL2	32	0x34	0xFFEC8034	32-bit
FlexCAN_2	MSG0_CS	32	0x80	0xFFEC8080	32-bit
FlexCAN_2	MSG0_ID	32	0x84	0xFFEC8084	32-bit
FlexCAN_2	MSG1_CS	32	0x90	0xFFEC8090	32-bit
FlexCAN_2	MSG1_ID	32	0x94	0xFFEC8094	32-bit
FlexCAN_2	MSG2_CS	32	0x00A0	0xFFEC80A0	32-bit
FlexCAN_2	MSG2_ID	32	0x00A4	0xFFEC80A4	32-bit
FlexCAN_2	MSG3_CS	32	0x00B0	0xFFEC80B0	32-bit
FlexCAN_2	MSG3_ID	32	0x00B4	0xFFEC80B4	32-bit
FlexCAN_2	MSG4_CS	32	0x00C0	0xFFEC80C0	32-bit
FlexCAN_2	MSG4_ID	32	0x00C4	0xFFEC80C4	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_2	MSG5_CS	32	0x00D0	0xFFEC80D0	32-bit
FlexCAN_2	MSG5_ID	32	0x00D4	0xFFEC80D4	32-bit
FlexCAN_2	MSG6_CS	32	0x00E0	0xFFEC80E0	32-bit
FlexCAN_2	MSG6_ID	32	0x00E4	0xFFEC80E4	32-bit
FlexCAN_2	MSG7_CS	32	0x00F0	0xFFEC80F0	32-bit
FlexCAN_2	MSG7_ID	32	0x00F4	0xFFEC80F4	32-bit
FlexCAN_2	MSG8_CS	32	0x100	0xFFEC8100	32-bit
FlexCAN_2	MSG8_ID	32	0x104	0xFFEC8104	32-bit
FlexCAN_2	MSG9_CS	32	0x110	0xFFEC8110	32-bit
FlexCAN_2	MSG9_ID	32	0x114	0xFFEC8114	32-bit
FlexCAN_2	MSG10_CS	32	0x120	0xFFEC8120	32-bit
FlexCAN_2	MSG10_ID	32	0x124	0xFFEC8124	32-bit
FlexCAN_2	MSG11_CS	32	0x130	0xFFEC8130	32-bit
FlexCAN_2	MSG11_ID	32	0x134	0xFFEC8134	32-bit
FlexCAN_2	MSG12_CS	32	0x140	0xFFEC8140	32-bit
FlexCAN_2	MSG12_ID	32	0x144	0xFFEC8144	32-bit
FlexCAN_2	MSG13_CS	32	0x150	0xFFEC8150	32-bit
FlexCAN_2	MSG13_ID	32	0x154	0xFFEC8154	32-bit
FlexCAN_2	MSG14_CS	32	0x160	0xFFEC8160	32-bit
FlexCAN_2	MSG14_ID	32	0x164	0xFFEC8164	32-bit
FlexCAN_2	MSG15_CS	32	0x170	0xFFEC8170	32-bit
FlexCAN_2	MSG15_ID	32	0x174	0xFFEC8174	32-bit
FlexCAN_2	MSG16_CS	32	0x180	0xFFEC8180	32-bit
FlexCAN_2	MSG16_ID	32	0x184	0xFFEC8184	32-bit
FlexCAN_2	MSG17_CS	32	0x190	0xFFEC8190	32-bit
FlexCAN_2	MSG17_ID	32	0x194	0xFFEC8194	32-bit
FlexCAN_2	MSG18_CS	32	0x01A0	0xFFEC81A0	32-bit
FlexCAN_2	MSG18_ID	32	0x01A4	0xFFEC81A4	32-bit
FlexCAN_2	MSG19_CS	32	0x01B0	0xFFEC81B0	32-bit
FlexCAN_2	MSG19_ID	32	0x01B4	0xFFEC81B4	32-bit
FlexCAN_2	MSG20_CS	32	0x01C0	0xFFEC81C0	32-bit
FlexCAN_2	MSG20_ID	32	0x01C4	0xFFEC81C4	32-bit
FlexCAN_2	MSG21_CS	32	0x01D0	0xFFEC81D0	32-bit
FlexCAN_2	MSG21_ID	32	0x01D4	0xFFEC81D4	32-bit
FlexCAN_2	MSG22_CS	32	0x01E0	0xFFEC81E0	32-bit
FlexCAN_2	MSG22_ID	32	0x01E4	0xFFEC81E4	32-bit
FlexCAN_2	MSG23_CS	32	0x01F0	0xFFEC81F0	32-bit
FlexCAN_2	MSG23_ID	32	0x01F4	0xFFEC81F4	32-bit
FlexCAN_2	MSG24_CS	32	0x200	0xFFEC8200	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_2	MSG24_ID	32	0x204	0xFFEC8204	32-bit
FlexCAN_2	MSG25_CS	32	0x210	0xFFEC8210	32-bit
FlexCAN_2	MSG25_ID	32	0x214	0xFFEC8214	32-bit
FlexCAN_2	MSG26_CS	32	0x220	0xFFEC8220	32-bit
FlexCAN_2	MSG26_ID	32	0x224	0xFFEC8224	32-bit
FlexCAN_2	MSG27_CS	32	0x230	0xFFEC8230	32-bit
FlexCAN_2	MSG27_ID	32	0x234	0xFFEC8234	32-bit
FlexCAN_2	MSG28_CS	32	0x240	0xFFEC8240	32-bit
FlexCAN_2	MSG28_ID	32	0x244	0xFFEC8244	32-bit
FlexCAN_2	MSG29_CS	32	0x250	0xFFEC8250	32-bit
FlexCAN_2	MSG29_ID	32	0x254	0xFFEC8254	32-bit
FlexCAN_2	MSG30_CS	32	0x260	0xFFEC8260	32-bit
FlexCAN_2	MSG30_ID	32	0x264	0xFFEC8264	32-bit
FlexCAN_2	MSG31_CS	32	0x270	0xFFEC8270	32-bit
FlexCAN_2	MSG31_ID	32	0x274	0xFFEC8274	32-bit
FlexCAN_2	MSG32_CS	32	0x280	0xFFEC8280	32-bit
FlexCAN_2	MSG32_ID	32	0x284	0xFFEC8284	32-bit
FlexCAN_2	MSG33_CS	32	0x290	0xFFEC8290	32-bit
FlexCAN_2	MSG33_ID	32	0x294	0xFFEC8294	32-bit
FlexCAN_2	MSG34_CS	32	0x2a0	0xFFEC82A0	32-bit
FlexCAN_2	MSG34_ID	32	0x2a4	0xFFEC82A4	32-bit
FlexCAN_2	MSG35_CS	32	0x2b0	0xFFEC82B0	32-bit
FlexCAN_2	MSG35_ID	32	0x2b4	0xFFEC82B4	32-bit
FlexCAN_2	MSG36_CS	32	0x2c0	0xFFEC82C0	32-bit
FlexCAN_2	MSG36_ID	32	0x2c4	0xFFEC82C4	32-bit
FlexCAN_2	MSG37_CS	32	0x2d0	0xFFEC82D0	32-bit
FlexCAN_2	MSG37_ID	32	0x2d4	0xFFEC82D4	32-bit
FlexCAN_2	MSG38_CS	32	0x2e0	0xFFEC82E0	32-bit
FlexCAN_2	MSG38_ID	32	0x2e4	0xFFEC82E4	32-bit
FlexCAN_2	MSG39_CS	32	0x2f0	0xFFEC82F0	32-bit
FlexCAN_2	MSG39_ID	32	0x2f4	0xFFEC82F4	32-bit
FlexCAN_2	MSG40_CS	32	0x300	0xFFEC8300	32-bit
FlexCAN_2	MSG40_ID	32	0x304	0xFFEC8304	32-bit
FlexCAN_2	MSG41_CS	32	0x310	0xFFEC8310	32-bit
FlexCAN_2	MSG41_ID	32	0x314	0xFFEC8314	32-bit
FlexCAN_2	MSG42_CS	32	0x320	0xFFEC8320	32-bit
FlexCAN_2	MSG42_ID	32	0x324	0xFFEC8324	32-bit
FlexCAN_2	MSG43_CS	32	0x330	0xFFEC8330	32-bit
FlexCAN_2	MSG43_ID	32	0x334	0xFFEC8334	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_2	MSG44_CS	32	0x340	0xFFEC8340	32-bit
FlexCAN_2	MSG44_ID	32	0x344	0xFFEC8344	32-bit
FlexCAN_2	MSG45_CS	32	0x350	0xFFEC8350	32-bit
FlexCAN_2	MSG45_ID	32	0x354	0xFFEC8354	32-bit
FlexCAN_2	MSG46_CS	32	0x360	0xFFEC8360	32-bit
FlexCAN_2	MSG46_ID	32	0x364	0xFFEC8364	32-bit
FlexCAN_2	MSG47_CS	32	0x370	0xFFEC8370	32-bit
FlexCAN_2	MSG47_ID	32	0x374	0xFFEC8374	32-bit
FlexCAN_2	MSG48_CS	32	0x380	0xFFEC8380	32-bit
FlexCAN_2	MSG48_ID	32	0x384	0xFFEC8384	32-bit
FlexCAN_2	MSG49_CS	32	0x390	0xFFEC8390	32-bit
FlexCAN_2	MSG49_ID	32	0x394	0xFFEC8394	32-bit
FlexCAN_2	MSG50_CS	32	0x3a0	0xFFEC83A0	32-bit
FlexCAN_2	MSG50_ID	32	0x3a4	0xFFEC83A4	32-bit
FlexCAN_2	MSG51_CS	32	0x3b0	0xFFEC83B0	32-bit
FlexCAN_2	MSG51_ID	32	0x3b4	0xFFEC83B4	32-bit
FlexCAN_2	MSG52_CS	32	0x3c0	0xFFEC83C0	32-bit
FlexCAN_2	MSG52_ID	32	0x3c4	0xFFEC83C4	32-bit
FlexCAN_2	MSG53_CS	32	0x3d0	0xFFEC83D0	32-bit
FlexCAN_2	MSG53_ID	32	0x3d4	0xFFEC83D4	32-bit
FlexCAN_2	MSG54_CS	32	0x3e0	0xFFEC83E0	32-bit
FlexCAN_2	MSG54_ID	32	0x3e4	0xFFEC83E4	32-bit
FlexCAN_2	MSG55_CS	32	0x3f0	0xFFEC83F0	32-bit
FlexCAN_2	MSG55_ID	32	0x3f4	0xFFEC83F4	32-bit
FlexCAN_2	MSG56_CS	32	0x400	0xFFEC8400	32-bit
FlexCAN_2	MSG56_ID	32	0x404	0xFFEC8404	32-bit
FlexCAN_2	MSG57_CS	32	0x410	0xFFEC8410	32-bit
FlexCAN_2	MSG57_ID	32	0x414	0xFFEC8414	32-bit
FlexCAN_2	MSG58_CS	32	0x420	0xFFEC8420	32-bit
FlexCAN_2	MSG58_ID	32	0x424	0xFFEC8424	32-bit
FlexCAN_2	MSG59_CS	32	0x430	0xFFEC8430	32-bit
FlexCAN_2	MSG59_ID	32	0x434	0xFFEC8434	32-bit
FlexCAN_2	MSG60_CS	32	0x440	0xFFEC8440	32-bit
FlexCAN_2	MSG60_ID	32	0x444	0xFFEC8444	32-bit
FlexCAN_2	MSG61_CS	32	0x450	0xFFEC8450	32-bit
FlexCAN_2	MSG61_ID	32	0x454	0xFFEC8454	32-bit
FlexCAN_2	MSG62_CS	32	0x460	0xFFEC8460	32-bit
FlexCAN_2	MSG62_ID	32	0x464	0xFFEC8464	32-bit
FlexCAN_2	MSG63_CS	32	0x470	0xFFEC8470	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexCAN_2	MSG63_ID	32	0x474	0xFFEC8474	32-bit
FlexCAN_2	RXIMR0	32	0x880	0xFFEC8880	32-bit
FlexCAN_2	RXIMR1	32	0x884	0xFFEC8884	32-bit
FlexCAN_2	RXIMR2	32	0x888	0xFFEC8888	32-bit
FlexCAN_2	RXIMR3	32	0x088C	0xFFEC888C	32-bit
FlexCAN_2	RXIMR4	32	0x890	0xFFEC8890	32-bit
FlexCAN_2	RXIMR5	32	0x894	0xFFEC8894	32-bit
FlexCAN_2	RXIMR6	32	0x898	0xFFEC8898	32-bit
FlexCAN_2	RXIMR7	32	0x089C	0xFFEC889C	32-bit
FlexCAN_2	RXIMR8	32	0x08A0	0xFFEC88A0	32-bit
FlexCAN_2	RXIMR9	32	0x08A4	0xFFEC88A4	32-bit
FlexCAN_2	RXIMR10	32	0x08A8	0xFFEC88A8	32-bit
FlexCAN_2	RXIMR11	32	0x08AC	0xFFEC88AC	32-bit
FlexCAN_2	RXIMR12	32	0x08B0	0xFFEC88B0	32-bit
FlexCAN_2	RXIMR13	32	0x08B4	0xFFEC88B4	32-bit
FlexCAN_2	RXIMR14	32	0x08B8	0xFFEC88B8	32-bit
FlexCAN_2	RXIMR15	32	0x08BC	0xFFEC88BC	32-bit
FlexCAN_2	RXIMR16	32	0x08C0	0xFFEC88C0	32-bit
FlexCAN_2	RXIMR17	32	0x08C4	0xFFEC88C4	32-bit
FlexCAN_2	RXIMR18	32	0x08C8	0xFFEC88C8	32-bit
FlexCAN_2	RXIMR19	32	0x08CC	0xFFEC88CC	32-bit
FlexCAN_2	RXIMR20	32	0x08D0	0xFFEC88D0	32-bit
FlexCAN_2	RXIMR21	32	0x08D4	0xFFEC88D4	32-bit
FlexCAN_2	RXIMR22	32	0x08D8	0xFFEC88D8	32-bit
FlexCAN_2	RXIMR23	32	0x08DC	0xFFEC88DC	32-bit
FlexCAN_2	RXIMR24	32	0x8e0	0xFFEC88E0	32-bit
FlexCAN_2	RXIMR25	32	0x8e4	0xFFEC88E4	32-bit
FlexCAN_2	RXIMR26	32	0x8e8	0xFFEC88E8	32-bit
FlexCAN_2	RXIMR27	32	0x08EC	0xFFEC88EC	32-bit
FlexCAN_2	RXIMR28	32	0x08F0	0xFFEC88F0	32-bit
FlexCAN_2	RXIMR29	32	0x08F4	0xFFEC88F4	32-bit
FlexCAN_2	RXIMR30	32	0x08F8	0xFFEC88F8	32-bit
FlexCAN_2	RXIMR31	32	0x08FC	0xFFEC88FC	32-bit
FlexCAN_2	FCMECR	32	0x0AE0	0xFFEC8AE0	32-bit
FlexPWM0	SUB0_INIT	16	0x2	0xFBC00002	16-bit
FlexPWM0	SUB0_CTRL2	16	0x4	0xFBC00004	16-bit
FlexPWM0	SUB0_CTRL1	16	0x6	0xFBC00006	16-bit
FlexPWM0	SUB0_VAL0	16	0x8	0xFBC00008	16-bit
FlexPWM0	SUB0_VAL1	16	0x000A	0xFBC0000A	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexPWM0	SUB0_VAL2	16	0x000C	0xFBC0000C	16-bit
FlexPWM0	SUB0_VAL3	16	0x000E	0xFBC0000E	16-bit
FlexPWM0	SUB0_VAL4	16	0x10	0xFBC00010	16-bit
FlexPWM0	SUB0_VAL5	16	0x12	0xFBC00012	16-bit
FlexPWM0	SUB0_OCTRL	16	0x18	0xFBC00018	16-bit
FlexPWM0	SUB0_INTEN	16	0x001C	0xFBC0001C	16-bit
FlexPWM0	SUB0_DMAEN	16	0x001E	0xFBC0001E	16-bit
FlexPWM0	SUB0_TCTRL	16	0x20	0xFBC00020	16-bit
FlexPWM0	SUB0_DISMAP	16	0x22	0xFBC00022	16-bit
FlexPWM0	SUB0_DTCNT0	16	0x24	0xFBC00024	16-bit
FlexPWM0	SUB0_DTCNT1	16	0x26	0xFBC00026	16-bit
FlexPWM0	SUB0_CAPTCTRLX	16	0x30	0xFBC00030	16-bit
FlexPWM0	SUB0_CAPTCMPX	16	0x32	0xFBC00032	16-bit
FlexPWM0	SUB1_INIT	16	0x52	0xFBC00052	16-bit
FlexPWM0	SUB1_CTRL2	16	0x54	0xFBC00054	16-bit
FlexPWM0	SUB1_CTRL1	16	0x56	0xFBC00056	16-bit
FlexPWM0	SUB1_VAL0	16	0x58	0xFBC00058	16-bit
FlexPWM0	SUB1_VAL1	16	0x005A	0xFBC0005A	16-bit
FlexPWM0	SUB1_VAL2	16	0x005C	0xFBC0005C	16-bit
FlexPWM0	SUB1_VAL3	16	0x005E	0xFBC0005E	16-bit
FlexPWM0	SUB1_VAL4	16	0x60	0xFBC00060	16-bit
FlexPWM0	SUB1_VAL5	16	0x62	0xFBC00062	16-bit
FlexPWM0	SUB1_OCTRL	16	0x68	0xFBC00068	16-bit
FlexPWM0	SUB1_INTEN	16	0x006C	0xFBC0006C	16-bit
FlexPWM0	SUB1_DMAEN	16	0x006E	0xFBC0006E	16-bit
FlexPWM0	SUB1_TCTRL	16	0x70	0xFBC00070	16-bit
FlexPWM0	SUB1_DISMAP	16	0x72	0xFBC00072	16-bit
FlexPWM0	SUB1_DTCNT0	16	0x74	0xFBC00074	16-bit
FlexPWM0	SUB1_DTCNT1	16	0x76	0xFBC00076	16-bit
FlexPWM0	SUB1_CAPTCTRLX	16	0x80	0xFBC00080	16-bit
FlexPWM0	SUB1_CAPTCMPX	16	0x82	0xFBC00082	16-bit
FlexPWM0	SUB2_INIT	16	0x00A2	0xFBC000A2	16-bit
FlexPWM0	SUB2_CTRL2	16	0x00A4	0xFBC000A4	16-bit
FlexPWM0	SUB2_CTRL1	16	0x00A6	0xFBC000A6	16-bit
FlexPWM0	SUB2_VAL0	16	0x00A8	0xFBC000A8	16-bit
FlexPWM0	SUB2_VAL1	16	0x00AA	0xFBC000AA	16-bit
FlexPWM0	SUB2_VAL2	16	0x00AC	0xFBC000AC	16-bit
FlexPWM0	SUB2_VAL3	16	0x00AE	0xFBC000AE	16-bit
FlexPWM0	SUB2_VAL4	16	0x00B0	0xFBC000B0	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexPWM0	SUB2_VAL5	16	0x00B2	0xFBC000B2	16-bit
FlexPWM0	SUB2_OCTRL	16	0x00B8	0xFBC000B8	16-bit
FlexPWM0	SUB2_INTEN	16	0x00BC	0xFBC000BC	16-bit
FlexPWM0	SUB2_DMAEN	16	0x00BE	0xFBC000BE	16-bit
FlexPWM0	SUB2_TCTRL	16	0x00C0	0xFBC000C0	16-bit
FlexPWM0	SUB2_DISMAP	16	0x00C2	0xFBC000C2	16-bit
FlexPWM0	SUB2_DTCNT0	16	0x00C4	0xFBC000C4	16-bit
FlexPWM0	SUB2_DTCNT1	16	0x00C6	0xFBC000C6	16-bit
FlexPWM0	SUB2_CAPTCTRLX	16	0x00D0	0xFBC000D0	16-bit
FlexPWM0	SUB2_CAPTCMPX	16	0x00D2	0xFBC000D2	16-bit
FlexPWM0	SUB3_INIT	16	0x00F2	0xFBC000F2	16-bit
FlexPWM0	SUB3_CTRL2	16	0x00F4	0xFBC000F4	16-bit
FlexPWM0	SUB3_CTRL1	16	0x00F6	0xFBC000F6	16-bit
FlexPWM0	SUB3_VAL0	16	0x00F8	0xFBC000F8	16-bit
FlexPWM0	SUB3_VAL1	16	0x00FA	0xFBC000FA	16-bit
FlexPWM0	SUB3_VAL2	16	0x00FC	0xFBC000FC	16-bit
FlexPWM0	SUB3_VAL3	16	0x00FE	0xFBC000FE	16-bit
FlexPWM0	SUB3_VAL4	16	0x100	0xFBC00100	16-bit
FlexPWM0	SUB3_VAL5	16	0x102	0xFBC00102	16-bit
FlexPWM0	SUB3_OCTRL	16	0x108	0xFBC00108	16-bit
FlexPWM0	SUB3_INTEN	16	0x10C	0xFBC0010C	16-bit
FlexPWM0	SUB3_DMAEN	16	0x10E	0xFBC0010E	16-bit
FlexPWM0	SUB3_TCTRL	16	0x110	0xFBC00110	16-bit
FlexPWM0	SUB3_DISMAP	16	0x112	0xFBC00112	16-bit
FlexPWM0	SUB3_DTCNT0	16	0x114	0xFBC00114	16-bit
FlexPWM0	SUB3_DTCNT1	16	0x116	0xFBC00116	16-bit
FlexPWM0	SUB3_CAPTCTRLX	16	0x120	0xFBC00120	16-bit
FlexPWM0	SUB3_CAPTCMPX	16	0x122	0xFBC00122	16-bit
FlexPWM0	OUTEN	16	0x140	0xFBC00140	16-bit
FlexPWM0	MASK	16	0x142	0xFBC00142	16-bit
FlexPWM0	SWCOUT	16	0x144	0xFBC00144	16-bit
FlexPWM0	DTSRCSEL	16	0x146	0xFBC00146	16-bit
FlexPWM0	MCTRL	16	0x148	0xFBC00148	16-bit
FlexPWM0	FCTRL	16	0x14C	0xFBC0014C	16-bit
FlexPWM1	SUB0_INIT	16	0x2	0xFBC00002	16-bit
FlexPWM1	SUB0_CTRL2	16	0x4	0xFBC00004	16-bit
FlexPWM1	SUB0_CTRL1	16	0x6	0xFBC00006	16-bit
FlexPWM1	SUB0_VAL0	16	0x8	0xFBC00008	16-bit
FlexPWM1	SUB0_VAL1	16	0x000A	0xFBC0000A	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexPWM1	SUB0_VAL2	16	0x000C	0xFBC0000C	16-bit
FlexPWM1	SUB0_VAL3	16	0x000E	0xFBC0000E	16-bit
FlexPWM1	SUB0_VAL4	16	0x10	0xFBC00010	16-bit
FlexPWM1	SUB0_VAL5	16	0x12	0xFBC00012	16-bit
FlexPWM1	SUB0_OCTRL	16	0x18	0xFBC00018	16-bit
FlexPWM1	SUB0_INTEN	16	0x001C	0xFBC0001C	16-bit
FlexPWM1	SUB0_DMAEN	16	0x001E	0xFBC0001E	16-bit
FlexPWM1	SUB0_TCTRL	16	0x20	0xFBC00020	16-bit
FlexPWM1	SUB0_DISMAP	16	0x22	0xFBC00022	16-bit
FlexPWM1	SUB0_DTCNT0	16	0x24	0xFBC00024	16-bit
FlexPWM1	SUB0_DTCNT1	16	0x26	0xFBC00026	16-bit
FlexPWM1	SUB0_CAPTCTRLX	16	0x30	0xFBC00030	16-bit
FlexPWM1	SUB0_CAPTCMPX	16	0x32	0xFBC00032	16-bit
FlexPWM1	SUB1_INIT	16	0x52	0xFBC00052	16-bit
FlexPWM1	SUB1_CTRL2	16	0x54	0xFBC00054	16-bit
FlexPWM1	SUB1_CTRL1	16	0x56	0xFBC00056	16-bit
FlexPWM1	SUB1_VAL0	16	0x58	0xFBC00058	16-bit
FlexPWM1	SUB1_VAL1	16	0x005A	0xFBC0005A	16-bit
FlexPWM1	SUB1_VAL2	16	0x005C	0xFBC0005C	16-bit
FlexPWM1	SUB1_VAL3	16	0x005E	0xFBC0005E	16-bit
FlexPWM1	SUB1_VAL4	16	0x60	0xFBC00060	16-bit
FlexPWM1	SUB1_VAL5	16	0x62	0xFBC00062	16-bit
FlexPWM1	SUB1_OCTRL	16	0x68	0xFBC00068	16-bit
FlexPWM1	SUB1_INTEN	16	0x006C	0xFBC0006C	16-bit
FlexPWM1	SUB1_DMAEN	16	0x006E	0xFBC0006E	16-bit
FlexPWM1	SUB1_TCTRL	16	0x70	0xFBC00070	16-bit
FlexPWM1	SUB1_DISMAP	16	0x72	0xFBC00072	16-bit
FlexPWM1	SUB1_DTCNT0	16	0x74	0xFBC00074	16-bit
FlexPWM1	SUB1_DTCNT1	16	0x76	0xFBC00076	16-bit
FlexPWM1	SUB1_CAPTCTRLX	16	0x80	0xFBC00080	16-bit
FlexPWM1	SUB1_CAPTCMPX	16	0x82	0xFBC00082	16-bit
FlexPWM1	SUB2_INIT	16	0x00A2	0xFBC000A2	16-bit
FlexPWM1	SUB2_CTRL2	16	0x00A4	0xFBC000A4	16-bit
FlexPWM1	SUB2_CTRL1	16	0x00A6	0xFBC000A6	16-bit
FlexPWM1	SUB2_VAL0	16	0x00A8	0xFBC000A8	16-bit
FlexPWM1	SUB2_VAL1	16	0x00AA	0xFBC000AA	16-bit
FlexPWM1	SUB2_VAL2	16	0x00AC	0xFBC000AC	16-bit
FlexPWM1	SUB2_VAL3	16	0x00AE	0xFBC000AE	16-bit
FlexPWM1	SUB2_VAL4	16	0x00B0	0xFBC000B0	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexPWM1	SUB2_VAL5	16	0x00B2	0xFBC000B2	16-bit
FlexPWM1	SUB2_OCTRL	16	0x00B8	0xFBC000B8	16-bit
FlexPWM1	SUB2_INTEN	16	0x00BC	0xFBC000BC	16-bit
FlexPWM1	SUB2_DMAEN	16	0x00BE	0xFBC000BE	16-bit
FlexPWM1	SUB2_TCTRL	16	0x00C0	0xFBC000C0	16-bit
FlexPWM1	SUB2_DISMAP	16	0x00C2	0xFBC000C2	16-bit
FlexPWM1	SUB2_DTCNT0	16	0x00C4	0xFBC000C4	16-bit
FlexPWM1	SUB2_DTCNT1	16	0x00C6	0xFBC000C6	16-bit
FlexPWM1	SUB2_CAPTCTRLX	16	0x00D0	0xFBC000D0	16-bit
FlexPWM1	SUB2_CAPTCMPX	16	0x00D2	0xFBC000D2	16-bit
FlexPWM1	SUB3_INIT	16	0x00F2	0xFBC000F2	16-bit
FlexPWM1	SUB3_CTRL2	16	0x00F4	0xFBC000F4	16-bit
FlexPWM1	SUB3_CTRL1	16	0x00F6	0xFBC000F6	16-bit
FlexPWM1	SUB3_VAL0	16	0x00F8	0xFBC000F8	16-bit
FlexPWM1	SUB3_VAL1	16	0x00FA	0xFBC000FA	16-bit
FlexPWM1	SUB3_VAL2	16	0x00FC	0xFBC000FC	16-bit
FlexPWM1	SUB3_VAL3	16	0x00FE	0xFBC000FE	16-bit
FlexPWM1	SUB3_VAL4	16	0x100	0xFBC00100	16-bit
FlexPWM1	SUB3_VAL5	16	0x102	0xFBC00102	16-bit
FlexPWM1	SUB3_OCTRL	16	0x108	0xFBC00108	16-bit
FlexPWM1	SUB3_INTEN	16	0x10C	0xFBC0010C	16-bit
FlexPWM1	SUB3_DMAEN	16	0x10E	0xFBC0010E	16-bit
FlexPWM1	SUB3_TCTRL	16	0x110	0xFBC00110	16-bit
FlexPWM1	SUB3_DISMAP	16	0x112	0xFBC00112	16-bit
FlexPWM1	SUB3_DTCNT0	16	0x114	0xFBC00114	16-bit
FlexPWM1	SUB3_DTCNT1	16	0x116	0xFBC00116	16-bit
FlexPWM1	SUB3_CAPTCTRLX	16	0x120	0xFBC00120	16-bit
FlexPWM1	SUB3_CAPTCMPX	16	0x122	0xFBC00122	16-bit
FlexPWM1	OUTEN	16	0x140	0xFBC00140	16-bit
FlexPWM1	MASK	16	0x142	0xFBC00142	16-bit
FlexPWM1	SWCOUT	16	0x144	0xFBC00144	16-bit
FlexPWM1	DTSRCSEL	16	0x146	0xFBC00146	16-bit
FlexPWM1	MCTRL	16	0x148	0xFBC00148	16-bit
FlexPWM1	FCTRL	16	0x14C	0xFBC0014C	16-bit
FlexRay	MCR	16	0x2	0xFFE50002	16-bit
FlexRay	SYMBADHR	16	0x4	0xFFE50004	16-bit
FlexRay	SYMBADLR	16	0x6	0xFFE50006	16-bit
FlexRay	STBSCR	16	0x8	0xFFE50008	16-bit
FlexRay	MBDSR	16	0xC	0xFFE5000C	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexRay	MBSSUTR	16	0xE	0xFFE5000E	16-bit
FlexRay	POCR	16	0x14	0xFFE50014	16-bit
FlexRay	GIFER	16	0x16	0xFFE50016	16-bit
FlexRay	PIER0	16	0x1C	0xFFE5001C	16-bit
FlexRay	PIER1	16	0x1E	0xFFE5001E	16-bit
FlexRay	SYMATOR	16	0x3E	0xFFE5003E	16-bit
FlexRay	SFTOR	16	0x42	0xFFE50042	16-bit
FlexRay	SFTCCSR	16	0x44	0xFFE50044	16-bit
FlexRay	SFIDRFR	16	0x46	0xFFE50046	16-bit
FlexRay	SFIDAFVR	16	0x48	0xFFE50048	16-bit
FlexRay	SFIDAFMR	16	0x4A	0xFFE5004A	16-bit
FlexRay	NMVLR	16	0x58	0xFFE50058	16-bit
FlexRay	TICCR	16	0x5A	0xFFE5005A	16-bit
FlexRay	TI1CYSR	16	0x5C	0xFFE5005C	16-bit
FlexRay	TI1MTOR	16	0x5E	0xFFE5005E	16-bit
FlexRay	TI2CR0	16	0x60	0xFFE50060	16-bit
FlexRay	TI2CR1	16	0x62	0xFFE50062	16-bit
FlexRay	MTSACFR	16	0x80	0xFFE50080	16-bit
FlexRay	MTSBCFR	16	0x82	0xFFE50082	16-bit
FlexRay	RFRFCTR	16	0x9A	0xFFE5009A	16-bit
FlexRay	PCR0	16	0xA0	0xFFE500A0	16-bit
FlexRay	PCR1	16	0xA2	0xFFE500A2	16-bit
FlexRay	PCR2	16	0xA4	0xFFE500A4	16-bit
FlexRay	PCR3	16	0xA6	0xFFE500A6	16-bit
FlexRay	PCR4	16	0xA8	0xFFE500A8	16-bit
FlexRay	PCR5	16	0xAA	0xFFE500AA	16-bit
FlexRay	PCR6	16	0xAC	0xFFE500AC	16-bit
FlexRay	PCR7	16	0xAE	0xFFE500AE	16-bit
FlexRay	PCR8	16	0xB0	0xFFE500B0	16-bit
FlexRay	PCR9	16	0xB2	0xFFE500B2	16-bit
FlexRay	PCR10	16	0xB4	0xFFE500B4	16-bit
FlexRay	PCR11	16	0xB6	0xFFE500B6	16-bit
FlexRay	PCR12	16	0xB8	0xFFE500B8	16-bit
FlexRay	PCR13	16	0xBA	0xFFE500BA	16-bit
FlexRay	PCR14	16	0xBC	0xFFE500BC	16-bit
FlexRay	PCR15	16	0xBE	0xFFE500BE	16-bit
FlexRay	PCR16	16	0xC0	0xFFE500C0	16-bit
FlexRay	PCR17	16	0xC2	0xFFE500C2	16-bit
FlexRay	PCR18	16	0xC4	0xFFE500C4	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexRay	PCR19	16	0xC6	0xFFE500C6	16-bit
FlexRay	PCR20	16	0xC8	0xFFE500C8	16-bit
FlexRay	PCR21	16	0xCA	0xFFE500CA	16-bit
FlexRay	PCR22	16	0xCC	0xFFE500CC	16-bit
FlexRay	PCR23	16	0xCE	0xFFE500CE	16-bit
FlexRay	PCR24	16	0xD0	0xFFE500D0	16-bit
FlexRay	PCR25	16	0xD2	0xFFE500D2	16-bit
FlexRay	PCR26	16	0xD4	0xFFE500D4	16-bit
FlexRay	PCR27	16	0xD6	0xFFE500D6	16-bit
FlexRay	PCR28	16	0xD8	0xFFE500D8	16-bit
FlexRay	PCR29	16	0xDA	0xFFE500DA	16-bit
FlexRay	PCR30	16	0xDC	0xFFE500DC	16-bit
FlexRay	MBCCFR0	16	0x802	0xFFE50802	16-bit
FlexRay	MBFIDR0	16	0x804	0xFFE50804	16-bit
FlexRay	MBCCFR1	16	0x80a	0xFFE5080A	16-bit
FlexRay	MBFIDR1	16	0x80c	0xFFE5080C	16-bit
FlexRay	MBCCFR2	16	0x812	0xFFE50812	16-bit
FlexRay	MBFIDR2	16	0x814	0xFFE50814	16-bit
FlexRay	MBCCFR3	16	0x81a	0xFFE5081A	16-bit
FlexRay	MBFIDR3	16	0x81c	0xFFE5081C	16-bit
FlexRay	MBCCFR4	16	0x822	0xFFE50822	16-bit
FlexRay	MBFIDR4	16	0x824	0xFFE50824	16-bit
FlexRay	MBCCFR5	16	0x82a	0xFFE5082A	16-bit
FlexRay	MBFIDR5	16	0x82c	0xFFE5082C	16-bit
FlexRay	MBCCFR6	16	0x832	0xFFE50832	16-bit
FlexRay	MBFIDR6	16	0x834	0xFFE50834	16-bit
FlexRay	MBCCFR7	16	0x83a	0xFFE5083A	16-bit
FlexRay	MBFIDR7	16	0x83c	0xFFE5083C	16-bit
FlexRay	MBCCFR8	16	0x842	0xFFE50842	16-bit
FlexRay	MBFIDR8	16	0x844	0xFFE50844	16-bit
FlexRay	MBCCFR9	16	0x84a	0xFFE5084A	16-bit
FlexRay	MBFIDR9	16	0x84c	0xFFE5084C	16-bit
FlexRay	MBCCFR10	16	0x852	0xFFE50852	16-bit
FlexRay	MBFIDR10	16	0x854	0xFFE50854	16-bit
FlexRay	MBCCFR11	16	0x85a	0xFFE5085A	16-bit
FlexRay	MBFIDR11	16	0x85c	0xFFE5085C	16-bit
FlexRay	MBCCFR12	16	0x862	0xFFE50862	16-bit
FlexRay	MBFIDR12	16	0x864	0xFFE50864	16-bit
FlexRay	MBCCFR13	16	0x86a	0xFFE5086A	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexRay	MBFIDR13	16	0x86c	0xFFE5086C	16-bit
FlexRay	MBCCFR14	16	0x872	0xFFE50872	16-bit
FlexRay	MBFIDR14	16	0x874	0xFFE50874	16-bit
FlexRay	MBCCFR15	16	0x87a	0xFFE5087A	16-bit
FlexRay	MBFIDR15	16	0x87c	0xFFE5087C	16-bit
FlexRay	MBCCFR16	16	0x882	0xFFE50882	16-bit
FlexRay	MBFIDR16	16	0x884	0xFFE50884	16-bit
FlexRay	MBCCFR17	16	0x88a	0xFFE5088A	16-bit
FlexRay	MBFIDR17	16	0x88c	0xFFE5088C	16-bit
FlexRay	MBCCFR18	16	0x892	0xFFE50892	16-bit
FlexRay	MBFIDR18	16	0x894	0xFFE50894	16-bit
FlexRay	MBCCFR19	16	0x89a	0xFFE5089A	16-bit
FlexRay	MBFIDR19	16	0x89c	0xFFE5089C	16-bit
FlexRay	MBCCFR20	16	0x8a2	0xFFE508A2	16-bit
FlexRay	MBFIDR20	16	0x8a4	0xFFE508A4	16-bit
FlexRay	MBCCFR21	16	0x8aa	0xFFE508AA	16-bit
FlexRay	MBFIDR21	16	0x8ac	0xFFE508AC	16-bit
FlexRay	MBCCFR22	16	0x8b2	0xFFE508B2	16-bit
FlexRay	MBFIDR22	16	0x8b4	0xFFE508B4	16-bit
FlexRay	MBCCFR23	16	0x8ba	0xFFE508BA	16-bit
FlexRay	MBFIDR23	16	0x8bc	0xFFE508BC	16-bit
FlexRay	MBCCFR24	16	0x8c2	0xFFE508C2	16-bit
FlexRay	MBFIDR24	16	0x8c4	0xFFE508C4	16-bit
FlexRay	MBCCFR25	16	0x8ca	0xFFE508CA	16-bit
FlexRay	MBFIDR25	16	0x8cc	0xFFE508CC	16-bit
FlexRay	MBCCFR26	16	0x8d2	0xFFE508D2	16-bit
FlexRay	MBFIDR26	16	0x8d4	0xFFE508D4	16-bit
FlexRay	MBCCFR27	16	0x8da	0xFFE508DA	16-bit
FlexRay	MBFIDR27	16	0x8dc	0xFFE508DC	16-bit
FlexRay	MBCCFR28	16	0x8E2	0xFFE508E2	16-bit
FlexRay	MBFIDR28	16	0x8E4	0xFFE508E4	16-bit
FlexRay	MBCCFR29	16	0x8ea	0xFFE508EA	16-bit
FlexRay	MBFIDR29	16	0x8ec	0xFFE508EC	16-bit
FlexRay	MBCCFR30	16	0x8f2	0xFFE508F2	16-bit
FlexRay	MBFIDR30	16	0x8f4	0xFFE508F4	16-bit
FlexRay	MBCCFR31	16	0x8fa	0xFFE508FA	16-bit
FlexRay	MBFIDR31	16	0x8fc	0xFFE508FC	16-bit
FlexRay	MBCCFR32	16	0x902	0xFFE50902	16-bit
FlexRay	MBFIDR32	16	0x904	0xFFE50904	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexRay	MBCCFR33	16	0x90a	0xFFE5090A	16-bit
FlexRay	MBFIDR33	16	0x90c	0xFFE5090C	16-bit
FlexRay	MBCCFR34	16	0x912	0xFFE50912	16-bit
FlexRay	MBFIDR34	16	0x914	0xFFE50914	16-bit
FlexRay	MBCCFR35	16	0x91a	0xFFE5091A	16-bit
FlexRay	MBFIDR35	16	0x91c	0xFFE5091C	16-bit
FlexRay	MBCCFR36	16	0x922	0xFFE50922	16-bit
FlexRay	MBFIDR36	16	0x924	0xFFE50924	16-bit
FlexRay	MBCCFR37	16	0x92a	0xFFE5092A	16-bit
FlexRay	MBFIDR37	16	0x92c	0xFFE5092C	16-bit
FlexRay	MBCCFR38	16	0x932	0xFFE50932	16-bit
FlexRay	MBFIDR38	16	0x934	0xFFE50934	16-bit
FlexRay	MBCCFR39	16	0x93a	0xFFE5093A	16-bit
FlexRay	MBFIDR39	16	0x93c	0xFFE5093C	16-bit
FlexRay	MBCCFR40	16	0x942	0xFFE50942	16-bit
FlexRay	MBFIDR40	16	0x944	0xFFE50944	16-bit
FlexRay	MBCCFR41	16	0x94a	0xFFE5094A	16-bit
FlexRay	MBFIDR41	16	0x94c	0xFFE5094C	16-bit
FlexRay	MBCCFR42	16	0x952	0xFFE50952	16-bit
FlexRay	MBFIDR42	16	0x954	0xFFE50954	16-bit
FlexRay	MBCCFR43	16	0x95a	0xFFE5095A	16-bit
FlexRay	MBFIDR43	16	0x95c	0xFFE5095C	16-bit
FlexRay	MBCCFR44	16	0x962	0xFFE50962	16-bit
FlexRay	MBFIDR44	16	0x964	0xFFE50964	16-bit
FlexRay	MBCCFR45	16	0x96a	0xFFE5096A	16-bit
FlexRay	MBFIDR45	16	0x96c	0xFFE5096C	16-bit
FlexRay	MBCCFR46	16	0x972	0xFFE50972	16-bit
FlexRay	MBFIDR46	16	0x974	0xFFE50974	16-bit
FlexRay	MBCCFR47	16	0x97a	0xFFE5097A	16-bit
FlexRay	MBFIDR47	16	0x97c	0xFFE5097C	16-bit
FlexRay	MBCCFR48	16	0x982	0xFFE50982	16-bit
FlexRay	MBFIDR48	16	0x984	0xFFE50984	16-bit
FlexRay	MBCCFR49	16	0x98a	0xFFE5098A	16-bit
FlexRay	MBFIDR49	16	0x98c	0xFFE5098C	16-bit
FlexRay	MBCCFR50	16	0x992	0xFFE50992	16-bit
FlexRay	MBFIDR50	16	0x994	0xFFE50994	16-bit
FlexRay	MBCCFR51	16	0x99a	0xFFE5099A	16-bit
FlexRay	MBFIDR51	16	0x99c	0xFFE5099C	16-bit
FlexRay	MBCCFR52	16	0x9a2	0xFFE509A2	16-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
FlexRay	MBFIDR52	16	0x9a4	0xFFE509A4	16-bit
FlexRay	MBCCFR53	16	0x9aa	0xFFE509AA	16-bit
FlexRay	MBFIDR53	16	0x9ac	0xFFE509AC	16-bit
FlexRay	MBCCFR54	16	0x9b2	0xFFE509B2	16-bit
FlexRay	MBFIDR54	16	0x9b4	0xFFE509B4	16-bit
FlexRay	MBCCFR55	16	0x9ba	0xFFE509BA	16-bit
FlexRay	MBFIDR55	16	0x9bc	0xFFE509BC	16-bit
FlexRay	MBCCFR56	16	0x9c2	0xFFE509C2	16-bit
FlexRay	MBFIDR56	16	0x9c4	0xFFE509C4	16-bit
FlexRay	MBCCFR57	16	0x9ca	0xFFE509CA	16-bit
FlexRay	MBFIDR57	16	0x9cc	0xFFE509CC	16-bit
FlexRay	MBCCFR58	16	0x9d2	0xFFE509D2	16-bit
FlexRay	MBFIDR58	16	0x9d4	0xFFE509D4	16-bit
FlexRay	MBCCFR59	16	0x9da	0xFFE509DA	16-bit
FlexRay	MBFIDR59	16	0x9dc	0xFFE509DC	16-bit
FlexRay	MBCCFR60	16	0x9E2	0xFFE509E2	16-bit
FlexRay	MBFIDR60	16	0x9E4	0xFFE509E4	16-bit
FlexRay	MBCCFR61	16	0x9ea	0xFFE509EA	16-bit
FlexRay	MBFIDR61	16	0x9ec	0xFFE509EC	16-bit
FlexRay	MBCCFR62	16	0x9f2	0xFFE509F2	16-bit
FlexRay	MBFIDR62	16	0x9f4	0xFFE509F4	16-bit
FlexRay	MBCCFR63	16	0x9fa	0xFFE509FA	16-bit
FlexRay	MBFIDR63	16	0x9fc	0xFFE509FC	16-bit
PLLDIG	PLL0CR	32	0x00	0xFFFB0100	32-bit
PLLDIG	PLL0DV	32	0x08	0xFFFB0108	32-bit
PLLDIG	PLL1CR	32	0x20	0xFFFB0120	32-bit
PLLDIG	PLL1DV	32	0x28	0xFFFB0128	32-bit
PLLDIG	PLL1FM	32	0x2C	0xFFFB012C	32-bit
PLLDIG	PLL1FD	32	0x30	0xFFFB0130	32-bit
LINFlex0	LINCR1	32	0x0	0xFBE84000	32-bit
LINFlex0	LINIER	32	0x4	0xFBE84004	32-bit
LINFlex0	UARTCR	32	0x10	0xFBE84010	32-bit
LINFlex0	LINTCSR	32	0x18	0xFBE84018	32-bit
LINFlex0	LINOCR	32	0x1c	0xFBE8401C	32-bit
LINFlex0	LINTOCR	32	0x20	0xFBE84020	32-bit
LINFlex0	LINFBRR	32	0x24	0xFBE84024	32-bit
LINFlex0	LINIBRR	32	0x28	0xFBE84028	32-bit
LINFlex0	LINCR2	32	0x30	0xFBE84030	32-bit
LINFlex0	BIDR	32	0x34	0xFBE84034	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
LINFlex0	IFER	32	0x40	0xFBE84040	32-bit
LINFlex0	IFMR	32	0x48	0xFBE84048	32-bit
LINFlex0	IFCR0	32	0x4C	0xFBE8404C	32-bit
LINFlex0	IFCR1	32	0x50	0xFBE84050	32-bit
LINFlex0	IFCR2	32	0x54	0xFBE84054	32-bit
LINFlex0	IFCR3	32	0x58	0xFBE84058	32-bit
LINFlex0	IFCR4	32	0x5C	0xFBE8405C	32-bit
LINFlex0	IFCR5	32	0x60	0xFBE84060	32-bit
LINFlex0	IFCR6	32	0x64	0xFBE84064	32-bit
LINFlex0	IFCR7	32	0x68	0xFBE84068	32-bit
LINFlex0	IFCR8	32	0x6C	0xFBE8406C	32-bit
LINFlex0	IFCR9	32	0x70	0xFBE84070	32-bit
LINFlex0	IFCR10	32	0x74	0xFBE84074	32-bit
LINFlex0	IFCR11	32	0x78	0xFBE84078	32-bit
LINFlex0	IFCR12	32	0x7C	0xFBE8407C	32-bit
LINFlex0	IFCR13	32	0x80	0xFBE84080	32-bit
LINFlex0	IFCR14	32	0x84	0xFBE84084	32-bit
LINFlex0	IFCR15	32	0x88	0xFBE84088	32-bit
LINFlex0	GCR	32	0x8c	0xFBE8408C	32-bit
LINFlex0	UARTPTO	32	0x90	0xFBE84090	32-bit
LINFlex0	DMATXE	32	0x98	0xFBE84098	32-bit
LINFlex0	DMARXE	32	0x9c	0xFBE8409C	32-bit
LINFlex1	LINCR1	32	0x0	0xFFE90000	32-bit
LINFlex1	LINIER	32	0x4	0xFFE90004	32-bit
LINFlex1	UARTCR	32	0x10	0xFFE90010	32-bit
LINFlex1	LINTCSR	32	0x18	0xFFE90018	32-bit
LINFlex1	LINOCR	32	0x1c	0xFFE9001C	32-bit
LINFlex1	LINTOCR	32	0x20	0xFFE90020	32-bit
LINFlex1	LINFBRR	32	0x24	0xFFE90024	32-bit
LINFlex1	LINIBRR	32	0x28	0xFFE90028	32-bit
LINFlex1	LINCR2	32	0x30	0xFFE90030	32-bit
LINFlex1	BIDR	32	0x34	0xFFE90034	32-bit
LINFlex1	IFER	32	0x40	0xFFE90040	32-bit
LINFlex1	IFMR	32	0x48	0xFFE90048	32-bit
LINFlex1	IFCR0	32	0x4C	0xFFE9004C	32-bit
LINFlex1	IFCR1	32	0x50	0xFFE90050	32-bit
LINFlex1	IFCR2	32	0x54	0xFFE90054	32-bit
LINFlex1	IFCR3	32	0x58	0xFFE90058	32-bit
LINFlex1	IFCR4	32	0x5C	0xFFE9005C	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
LINFlex1	IFCR5	32	0x60	0xFFE90060	32-bit
LINFlex1	IFCR6	32	0x64	0xFFE90064	32-bit
LINFlex1	IFCR7	32	0x68	0xFFE90068	32-bit
LINFlex1	IFCR8	32	0x6C	0xFFE9006C	32-bit
LINFlex1	IFCR9	32	0x70	0xFFE90070	32-bit
LINFlex1	IFCR10	32	0x74	0xFFE90074	32-bit
LINFlex1	IFCR11	32	0x78	0xFFE90078	32-bit
LINFlex1	IFCR12	32	0x7C	0xFFE9007C	32-bit
LINFlex1	IFCR13	32	0x80	0xFFE90080	32-bit
LINFlex1	IFCR14	32	0x84	0xFFE90084	32-bit
LINFlex1	IFCR15	32	0x88	0xFFE90088	32-bit
LINFlex1	GCR	32	0x8c	0xFFE9008C	32-bit
LINFlex1	UARTPTO	32	0x90	0xFFE90090	32-bit
LINFlex1	DMATXE	32	0x98	0xFFE90098	32-bit
LINFlex1	DMARXE	32	0x9c	0xFFE9009C	32-bit
MC_CGM	PCS_SDUR	8	0x700	0xFFFB0700	8-bit
MC_CGM	PCS_DIVC1	32	0x704	0xFFFB0704	32-bit
MC_CGM	PCS_DIVS1	32	0x708	0xFFFB0708	32-bit
MC_CGM	PCS_DIVE1	32	0x70C	0xFFFB070C	32-bit
MC_CGM	PCS_DIVC2	32	0x710	0xFFFB0710	32-bit
MC_CGM	PCS_DIVS2	32	0x714	0xFFFB0714	32-bit
MC_CGM	PCS_DIVE2	32	0x718	0xFFFB0718	32-bit
MC_CGM	PCS_DIVC4	32	0x728	0xFFFB0728	32-bit
MC_CGM	PCS_DIVS4	32	0x72C	0xFFFB072C	32-bit
MC_CGM	PCS_DIVE4	32	0x730	0xFFFB0730	32-bit
MC_CGM	SC_DC0	32	0x7E8	0xFFFB07E8	32-bit
MC_CGM	AC0_SC	32	0x800	0xFFFB0800	32-bit
MC_CGM	AC0_DC0	32	0x808	0xFFFB0808	32-bit
MC_CGM	AC0_DC1	32	0x80C	0xFFFB080C	32-bit
MC_CGM	AC0_DC2	32	0x810	0xFFFB0810	32-bit
MC_CGM	AC1_DC0	32	0x828	0xFFFB0828	32-bit
MC_CGM	AC1_DC1	32	0x82C	0xFFFB082C	32-bit
MC_CGM	AC2_DC0	32	0x848	0xFFFB0848	32-bit
MC_CGM	AC3_SC	32	0x860	0xFFFB0860	32-bit
MC_CGM	AC4_SC	32	0x880	0xFFFB0880	32-bit
MC_CGM	AC5_SC	32	0x8A0	0xFFFB08A0	32-bit
MC_CGM	AC5_DC0	32	0x8A8	0xFFFB08A8	32-bit
MC_CGM	AC6_SC	32	0x8C0	0xFFFB08C0	32-bit
MC_CGM	AC6_DC0	32	0x8C8	0xFFFB08C8	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
MC_ME	ME	32	0x8	0xFFFB8008	32-bit
MC_ME	IM	32	0x10	0xFFFB8010	32-bit
MC_ME	TEST_MC	32	0x24	0xFFFB8024	32-bit
MC_ME	SAFE_MC	32	0x28	0xFFFB8028	32-bit
MC_ME	DRUN_MC	32	0x02C	0xFFFB802C	32-bit
MC_ME	RUN0_MC	32	0x30	0xFFFB8030	32-bit
MC_ME	RUN1_MC	32	0x34	0xFFFB8034	32-bit
MC_ME	RUN2_MC	32	0x38	0xFFFB8038	32-bit
MC_ME	RUN3_MC	32	0x03C	0xFFFB803C	32-bit
MC_ME	HALT0_MC	32	0x40	0xFFFB8040	32-bit
MC_ME	STOP0_MC	32	0x48	0xFFFB8048	32-bit
MC_ME	RUN_PC0	32	0x80	0xFFFB8080	32-bit
MC_ME	RUN_PC1	32	0x84	0xFFFB8084	32-bit
MC_ME	RUN_PC2	32	0x88	0xFFFB8088	32-bit
MC_ME	RUN_PC3	32	0x08C	0xFFFB808C	32-bit
MC_ME	RUN_PC4	32	0x90	0xFFFB8090	32-bit
MC_ME	RUN_PC5	32	0x94	0xFFFB8094	32-bit
MC_ME	RUN_PC6	32	0x98	0xFFFB8098	32-bit
MC_ME	RUN_PC7	32	0x09C	0xFFFB809C	32-bit
MC_ME	LP_PC0	32	0x0A0	0xFFFB80A0	32-bit
MC_ME	LP_PC1	32	0x0A4	0xFFFB80A4	32-bit
MC_ME	LP_PC2	32	0x0A8	0xFFFB80A8	32-bit
MC_ME	LP_PC3	32	0x0AC	0xFFFB80AC	32-bit
MC_ME	LP_PC4	32	0x0B0	0xFFFB80B0	32-bit
MC_ME	LP_PC5	32	0x0B4	0xFFFB80B4	32-bit
MC_ME	LP_PC6	32	0x0B8	0xFFFB80B8	32-bit
MC_ME	LP_PC7	32	0x0BC	0xFFFB80BC	32-bit
MC_ME	PCTL9	8	0xC9	0xFFFB80C9	8-bit
MC_ME	PCTL11	8	0xCB	0xFFFB80CB	8-bit
MC_ME	PCTL30	8	0xDE	0xFFFB80DE	8-bit
MC_ME	PCTL36	8	0xE4	0xFFFB80E4	8-bit
MC_ME	PCTL38	8	0xE6	0xFFFB80E6	8-bit
MC_ME	PCTL77	8	0x10D	0xFFFB810D	8-bit
MC_ME	PCTL78	8	0x10E	0xFFFB810E	8-bit
MC_ME	PCTL79	8	0x10F	0xFFFB810F	8-bit
MC_ME	PCTL91	8	0x11B	0xFFFB811B	8-bit
MC_ME	PCTL98	8	0x122	0xFFFB8122	8-bit
MC_ME	PCTL99	8	0x123	0xFFFB8123	8-bit
MC_ME	PCTL104	8	0x128	0xFFFB8128	8-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
MC_ME	PCTL107	8	0x12B	0xFFFB812B	8-bit
MC_ME	PCTL124	8	0x13C	0xFFFB813C	8-bit
MC_ME	PCTL126	8	0x13E	0xFFFB813E	8-bit
MC_ME	PCTL137	8	0x149	0xFFFB8149	8-bit
MC_ME	PCTL141	8	0x14D	0xFFFB814D	8-bit
MC_ME	PCTL144	8	0x150	0xFFFB8150	8-bit
MC_ME	PCTL146	8	0x152	0xFFFB8152	8-bit
MC_ME	PCTL204	8	0x18C	0xFFFB818C	8-bit
MC_ME	PCTL208	8	0x190	0xFFFB8190	8-bit
MC_ME	PCTL209	8	0x191	0xFFFB8191	8-bit
MC_ME	PCTL214	8	0x196	0xFFFB8196	8-bit
MC_ME	PCTL235	8	0x1AB	0xFFFB81AB	8-bit
MC_ME	PCTL237	8	0x1AD	0xFFFB81AD	8-bit
MC_ME	PCTL239	8	0x1AF	0xFFFB81AF	8-bit
MC_ME	PCTL245	8	0x1B5	0xFFFB81B5	8-bit
MC_ME	PCTL247	8	0x1B7	0xFFFB81B7	8-bit
MC_ME	PCTL251	8	0x1BB	0xFFFB81BB	8-bit
MC_ME	PCTL255	8	0x1BF	0xFFFB81BF	8-bit
MC_RGM	DERD	32	0x10	0xFFFA8010	32-bit
MC_RGM	DEAR	32	0x20	0xFFFA8020	32-bit
MC_RGM	DBRE	32	0x30	0xFFFA8030	32-bit
MC_RGM	FERD	32	0x310	0xFFFA8310	32-bit
MC_RGM	FEAR	32	0x320	0xFFFA8320	32-bit
MC_RGM	FBRE	32	0x330	0xFFFA8330	32-bit
MC_RGM	FESS	32	0x340	0xFFFA8340	32-bit
MC_RGM	FRETR	8	0x604	0xFFFA8604	8-bit
MEMU	CTRL	32	0x0	0xFFF50000	32-bit
MEMU	DEBUG	32	0xC	0xFFF5000C	32-bit
PIT	PITMCR	32	0x0	0xFFF84000	32-bit
PIT	LDVAL0	32	0x100	0xFFF84100	32-bit
PIT	TCTRL0	32	0x108	0xFFF84108	32-bit
PIT	LDVAL1	32	0x110	0xFFF84110	32-bit
PIT	TCTRL1	32	0x118	0xFFF84118	32-bit
PIT	LDVAL2	32	0x120	0xFFF84120	32-bit
PIT	TCTRL2	32	0x128	0xFFF84128	32-bit
PIT	LDVAL3	32	0x130	0xFFF84130	32-bit
PIT	TCTRL3	32	0x138	0xFFF84138	32-bit
PMC	PMCCR	32	0x8	0xFFFA0408	32-bit
PMC	IER	32	0x10	0xFFFA0410	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
PMC	REE_0	32	0x30	0xFFFA0430	32-bit
PMC	RES_0	32	0x40	0xFFFA0440	32-bit
PMC	ESR_TD	32	0x100	0xFFFA0500	32-bit
PMC	REE_TD	32	0x104	0xFFFA0504	32-bit
PMC	RES_TD	32	0x108	0xFFFA0508	32-bit
PMC	CTL_TD	32	0x10c	0xFFFA050C	32-bit
PMC	VD_UTST	32	0x140	0xFFFA0540	32-bit
IRCOSC	CTL	32	0x0	0xFFFB0000	32-bit
SRX_0	GBL_CTRL	32	0x0	0xFFE5C000	32-bit
SRX_0	CHNL_EN	32	0x4	0xFFE5C004	32-bit
SRX_0	DATA_CTRL1	32	0x18	0xFFE5C018	32-bit
SRX_0	FDMA_CTRL	32	0x28	0xFFE5C028	32-bit
SRX_0	SDMA_CTRL	32	0x2c	0xFFE5C02C	32-bit
SRX_0	FRDY_IE	32	0x34	0xFFE5C034	32-bit
SRX_0	SRDY_IE	32	0x38	0xFFE5C038	32-bit
SRX_0	CH0_CLK_CTRL	32	0x60	0xFFE5C060	32-bit
SRX_0	CH0_CONFIG	32	0x68	0xFFE5C068	32-bit
SRX_0	CH1_CLK_CTRL	32	0x70	0xFFE5C070	32-bit
SRX_0	CH1_CONFIG	32	0x78	0xFFE5C078	32-bit
SRX_1	GBL_CTRL	32	0x0	0xFBE5C000	32-bit
SRX_1	CHNL_EN	32	0x4	0xFBE5C004	32-bit
SRX_1	DATA_CTRL1	32	0x18	0xFBE5C018	32-bit
SRX_1	FDMA_CTRL	32	0x28	0xFBE5C028	32-bit
SRX_1	SDMA_CTRL	32	0x2c	0xFBE5C02C	32-bit
SRX_1	FRDY_IE	32	0x34	0xFBE5C034	32-bit
SRX_1	SRDY_IE	32	0x38	0xFBE5C038	32-bit
SRX_1	CH0_CLK_CTRL	32	0x60	0xFBE5C060	32-bit
SRX_1	CH0_CONFIG	32	0x68	0xFBE5C068	32-bit
SRX_1	CH1_CLK_CTRL	32	0x70	0xFBE5C070	32-bit
SRX_1	CH1_CONFIG	32	0x78	0xFBE5C078	32-bit
SIPI	CCR0	32	0x00	0xFFFD0000	32-bit
SIPI	CIR0	32	0x0C	0xFFFD000C	32-bit
SIPI	CTOR0	32	0x10	0xFFFD0010	32-bit
SIPI	CCR1	32	0x20	0xFFFD0020	32-bit
SIPI	CIR1	32	0x2C	0xFFFD002C	32-bit
SIPI	CTOR1	32	0x30	0xFFFD0030	32-bit
SIPI	CCR2	32	0x40	0xFFFD0040	32-bit
SIPI	CIR2	32	0x4C	0xFFFD004C	32-bit
SIPI	CTOR2	32	0x50	0xFFFD0050	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
SIPI	CCR3	32	0x7C	0xFFFD007C	32-bit
SIPI	CIR3	32	0x88	0xFFFD0088	32-bit
SIPI	CTOR3	32	0x8C	0xFFFD008C	32-bit
SIPI	SIPIMCR	32	0x9C	0xFFFD009C	32-bit
SGEN	CTRL	32	0x0	0xFBC40000	32-bit
SIUL2	DIRER0	32	0x18	0xFFFC0018	32-bit
SIUL2	DIRSR0	32	0x20	0xFFFC0020	32-bit
SIUL2	IREER0	32	0x28	0xFFFC0028	32-bit
SIUL2	IFEER0	32	0x30	0xFFFC0030	32-bit
SIUL2	IFER0	32	0x38	0xFFFC0038	32-bit
SIUL2	IFMCR0	32	0x40	0xFFFC0040	32-bit
SIUL2	IFMCR1	32	0x44	0xFFFC0044	32-bit
SIUL2	IFMCR2	32	0x48	0xFFFC0048	32-bit
SIUL2	IFMCR3	32	0x4c	0xFFFC004C	32-bit
SIUL2	IFMCR4	32	0x50	0xFFFC0050	32-bit
SIUL2	IFMCR5	32	0x54	0xFFFC0054	32-bit
SIUL2	IFMCR6	32	0x58	0xFFFC0058	32-bit
SIUL2	IFMCR7	32	0x5c	0xFFFC005C	32-bit
SIUL2	IFMCR8	32	0x60	0xFFFC0060	32-bit
SIUL2	IFMCR9	32	0x64	0xFFFC0064	32-bit
SIUL2	IFMCR10	32	0x68	0xFFFC0068	32-bit
SIUL2	IFMCR11	32	0x6c	0xFFFC006C	32-bit
SIUL2	IFMCR12	32	0x70	0xFFFC0070	32-bit
SIUL2	IFMCR13	32	0x74	0xFFFC0074	32-bit
SIUL2	IFMCR14	32	0x78	0xFFFC0078	32-bit
SIUL2	IFMCR15	32	0x7c	0xFFFC007C	32-bit
SIUL2	IFMCR16	32	0x80	0xFFFC0080	32-bit
SIUL2	IFMCR17	32	0x84	0xFFFC0084	32-bit
SIUL2	IFMCR18	32	0x88	0xFFFC0088	32-bit
SIUL2	IFMCR19	32	0x8c	0xFFFC008C	32-bit
SIUL2	IFMCR20	32	0x90	0xFFFC0090	32-bit
SIUL2	IFMCR21	32	0x94	0xFFFC0094	32-bit
SIUL2	IFMCR22	32	0x98	0xFFFC0098	32-bit
SIUL2	IFMCR23	32	0x9c	0xFFFC009C	32-bit
SIUL2	IFMCR24	32	0xa0	0xFFFC00A0	32-bit
SIUL2	IFMCR25	32	0xa4	0xFFFC00A4	32-bit
SIUL2	IFMCR26	32	0xa8	0xFFFC00A8	32-bit
SIUL2	IFMCR27	32	0xac	0xFFFC00AC	32-bit
SIUL2	IFMCR28	32	0xb0	0xFFFC00B0	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
SIUL2	IFMCR29	32	0xb4	0xFFFC00B4	32-bit
SIUL2	IFMCR30	32	0xb8	0xFFFC00B8	32-bit
SIUL2	IFMCR31	32	0xbc	0xFFFC00BC	32-bit
SIUL2	IFCPR	32	0xc0	0xFFFC00C0	32-bit
SIUL2	MSCR0	32	0x240	0xFFFC0240	32-bit
SIUL2	MSCR1	32	0x244	0xFFFC0244	32-bit
SIUL2	MSCR2	32	0x248	0xFFFC0248	32-bit
SIUL2	MSCR3	32	0x24C	0xFFFC024C	32-bit
SIUL2	MSCR4	32	0x250	0xFFFC0250	32-bit
SIUL2	MSCR5	32	0x254	0xFFFC0254	32-bit
SIUL2	MSCR6	32	0x258	0xFFFC0258	32-bit
SIUL2	MSCR7	32	0x25C	0xFFFC025C	32-bit
SIUL2	MSCR8	32	0x260	0xFFFC0260	32-bit
SIUL2	MSCR9	32	0x264	0xFFFC0264	32-bit
SIUL2	MSCR10	32	0x268	0xFFFC0268	32-bit
SIUL2	MSCR11	32	0x26C	0xFFFC026C	32-bit
SIUL2	MSCR12	32	0x270	0xFFFC0270	32-bit
SIUL2	MSCR13	32	0x274	0xFFFC0274	32-bit
SIUL2	MSCR14	32	0x278	0xFFFC0278	32-bit
SIUL2	MSCR15	32	0x27C	0xFFFC027C	32-bit
SIUL2	MSCR16	32	0x280	0xFFFC0280	32-bit
SIUL2	MSCR17	32	0x284	0xFFFC0284	32-bit
SIUL2	MSCR18	32	0x288	0xFFFC0288	32-bit
SIUL2	MSCR19	32	0x28C	0xFFFC028C	32-bit
SIUL2	MSCR20	32	0x290	0xFFFC0290	32-bit
SIUL2	MSCR21	32	0x294	0xFFFC0294	32-bit
SIUL2	MSCR22	32	0x298	0xFFFC0298	32-bit
SIUL2	MSCR23	32	0x29C	0xFFFC029C	32-bit
SIUL2	MSCR24	32	0x2A0	0xFFFC02A0	32-bit
SIUL2	MSCR25	32	0x2A4	0xFFFC02A4	32-bit
SIUL2	MSCR26	32	0x2A8	0xFFFC02A8	32-bit
SIUL2	MSCR27	32	0x2AC	0xFFFC02AC	32-bit
SIUL2	MSCR28	32	0x2B0	0xFFFC02B0	32-bit
SIUL2	MSCR29	32	0x2B4	0xFFFC02B4	32-bit
SIUL2	MSCR30	32	0x2B8	0xFFFC02B8	32-bit
SIUL2	MSCR31	32	0x2BC	0xFFFC02BC	32-bit
SIUL2	MSCR32	32	0x2C0	0xFFFC02C0	32-bit
SIUL2	MSCR33	32	0x2C4	0xFFFC02C4	32-bit
SIUL2	MSCR34	32	0x2C8	0xFFFC02C8	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
SIUL2	MSCR36	32	0x2D0	0xFFFC02D0	32-bit
SIUL2	MSCR37	32	0x2D4	0xFFFC02D4	32-bit
SIUL2	MSCR38	32	0x2D8	0xFFFC02D8	32-bit
SIUL2	MSCR39	32	0x2DC	0xFFFC02DC	32-bit
SIUL2	MSCR42	32	0x2E8	0xFFFC02E8	32-bit
SIUL2	MSCR43	32	0x2EC	0xFFFC02EC	32-bit
SIUL2	MSCR44	32	0x2F0	0xFFFC02F0	32-bit
SIUL2	MSCR45	32	0x2F4	0xFFFC02F4	32-bit
SIUL2	MSCR46	32	0x2F8	0xFFFC02F8	32-bit
SIUL2	MSCR47	32	0x2FC	0xFFFC02FC	32-bit
SIUL2	MSCR48	32	0x300	0xFFFC0300	32-bit
SIUL2	MSCR49	32	0x304	0xFFFC0304	32-bit
SIUL2	MSCR50	32	0x308	0xFFFC0308	32-bit
SIUL2	MSCR51	32	0x30C	0xFFFC030C	32-bit
SIUL2	MSCR52	32	0x310	0xFFFC0310	32-bit
SIUL2	MSCR53	32	0x314	0xFFFC0314	32-bit
SIUL2	MSCR54	32	0x318	0xFFFC0318	32-bit
SIUL2	MSCR55	32	0x31C	0xFFFC031C	32-bit
SIUL2	MSCR56	32	0x320	0xFFFC0320	32-bit
SIUL2	MSCR57	32	0x324	0xFFFC0324	32-bit
SIUL2	MSCR58	32	0x328	0xFFFC0328	32-bit
SIUL2	MSCR59	32	0x32C	0xFFFC032C	32-bit
SIUL2	MSCR60	32	0x330	0xFFFC0330	32-bit
SIUL2	MSCR62	32	0x338	0xFFFC0338	32-bit
SIUL2	MSCR64	32	0x340	0xFFFC0340	32-bit
SIUL2	MSCR66	32	0x348	0xFFFC0348	32-bit
SIUL2	MSCR68	32	0x350	0xFFFC0350	32-bit
SIUL2	MSCR69	32	0x354	0xFFFC0354	32-bit
SIUL2	MSCR70	32	0x358	0xFFFC0358	32-bit
SIUL2	MSCR71	32	0x35C	0xFFFC035C	32-bit
SIUL2	MSCR73	32	0x364	0xFFFC0364	32-bit
SIUL2	MSCR74	32	0x368	0xFFFC0368	32-bit
SIUL2	MSCR75	32	0x36C	0xFFFC036C	32-bit
SIUL2	MSCR76	32	0x370	0xFFFC0370	32-bit
SIUL2	MSCR77	32	0x374	0xFFFC0374	32-bit
SIUL2	MSCR78	32	0x378	0xFFFC0378	32-bit
SIUL2	MSCR79	32	0x37C	0xFFFC037C	32-bit
SIUL2	MSCR80	32	0x380	0xFFFC0380	32-bit
SIUL2	MSCR83	32	0x38C	0xFFFC038C	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
SIUL2	MSCR84	32	0x390	0xFFFC0390	32-bit
SIUL2	MSCR85	32	0x394	0xFFFC0394	32-bit
SIUL2	MSCR86	32	0x398	0xFFFC0398	32-bit
SIUL2	MSCR87	32	0x39C	0xFFFC039C	32-bit
SIUL2	MSCR88	32	0x3A0	0xFFFC03A0	32-bit
SIUL2	MSCR89	32	0x3A4	0xFFFC03A4	32-bit
SIUL2	MSCR90	32	0x3A8	0xFFFC03A8	32-bit
SIUL2	MSCR91	32	0x3AC	0xFFFC03AC	32-bit
SIUL2	MSCR92	32	0x3B0	0xFFFC03B0	32-bit
SIUL2	MSCR93	32	0x3B4	0xFFFC03B4	32-bit
SIUL2	MSCR94	32	0x3B8	0xFFFC03B8	32-bit
SIUL2	MSCR95	32	0x3BC	0xFFFC03BC	32-bit
SIUL2	MSCR98	32	0x3C8	0xFFFC03C8	32-bit
SIUL2	MSCR99	32	0x3CC	0xFFFC03CC	32-bit
SIUL2	MSCR100	32	0x3D0	0xFFFC03D0	32-bit
SIUL2	MSCR101	32	0x3D4	0xFFFC03D4	32-bit
SIUL2	MSCR102	32	0x3D8	0xFFFC03D8	32-bit
SIUL2	MSCR103	32	0x3DC	0xFFFC03DC	32-bit
SIUL2	MSCR104	32	0x3E0	0xFFFC03E0	32-bit
SIUL2	MSCR105	32	0x3E4	0xFFFC03E4	32-bit
SIUL2	MSCR106	32	0x3E8	0xFFFC03E8	32-bit
SIUL2	MSCR107	32	0x3EC	0xFFFC03EC	32-bit
SIUL2	MSCR116	32	0x410	0xFFFC0410	32-bit
SIUL2	MSCR117	32	0x414	0xFFFC0414	32-bit
SIUL2	MSCR118	32	0x418	0xFFFC0418	32-bit
SIUL2	MSCR119	32	0x41C	0xFFFC041C	32-bit
SIUL2	MSCR120	32	0x420	0xFFFC0420	32-bit
SIUL2	MSCR121	32	0x424	0xFFFC0424	32-bit
SIUL2	MSCR122	32	0x428	0xFFFC0428	32-bit
SIUL2	MSCR123	32	0x42C	0xFFFC042C	32-bit
SIUL2	MSCR124	32	0x430	0xFFFC0430	32-bit
SIUL2	MSCR125	32	0x434	0xFFFC0434	32-bit
SIUL2	MSCR126	32	0x438	0xFFFC0438	32-bit
SIUL2	MSCR127	32	0x43C	0xFFFC043C	32-bit
SIUL2	MSCR128	32	0x440	0xFFFC0440	32-bit
SIUL2	MSCR129	32	0x444	0xFFFC0444	32-bit
SIUL2	MSCR130	32	0x448	0xFFFC0448	32-bit
SIUL2	MSCR131	32	0x44C	0xFFFC044C	32-bit
SIUL2	MSCR132	32	0x450	0xFFFC0450	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
SIUL2	MSCR133	32	0x454	0xFFFC0454	32-bit
SIUL2	MSCR134	32	0x458	0xFFFC0458	32-bit
SIUL2	MSCR135	32	0x45C	0xFFFC045C	32-bit
SIUL2	MSCR136	32	0x460	0xFFFC0460	32-bit
SIUL2	MSCR137	32	0x464	0xFFFC0464	32-bit
SIUL2	MSCR138	32	0x468	0xFFFC0468	32-bit
SIUL2	MSCR139	32	0x46C	0xFFFC046C	32-bit
SIUL2	MSCR140	32	0x470	0xFFFC0470	32-bit
SIUL2	MSCR141	32	0x474	0xFFFC0474	32-bit
SIUL2	MSCR142	32	0x478	0xFFFC0478	32-bit
SIUL2	MSCR143	32	0x47C	0xFFFC047C	32-bit
SIUL2	MSCR144	32	0x480	0xFFFC0480	32-bit
SIUL2	MSCR145	32	0x484	0xFFFC0484	32-bit
SIUL2	MSCR146	32	0x488	0xFFFC0488	32-bit
SIUL2	MSCR147	32	0x48C	0xFFFC048C	32-bit
SIUL2	MSCR148	32	0x490	0xFFFC0490	32-bit
SIUL2	MSCR149	32	0x494	0xFFFC0494	32-bit
SIUL2	MSCR150	32	0x498	0xFFFC0498	32-bit
SIUL2	MSCR151	32	0x49C	0xFFFC049C	32-bit
SIUL2	MSCR152	32	0x4A0	0xFFFC04A0	32-bit
SIUL2	MSCR153	32	0x4A4	0xFFFC04A4	32-bit
SIUL2	MSCR154	32	0x4A8	0xFFFC04A8	32-bit
SIUL2	IMCR32	32	0xAC0	0xFFFC0AC0	32-bit
SIUL2	IMCR33	32	0xAC4	0xFFFC0AC4	32-bit
SIUL2	IMCR34	32	0xAC8	0xFFFC0AC8	32-bit
SIUL2	IMCR38	32	0xAD8	0xFFFC0AD8	32-bit
SIUL2	IMCR39	32	0xADC	0xFFFC0ADC	32-bit
SIUL2	IMCR41	32	0xAE4	0xFFFC0AE4	32-bit
SIUL2	IMCR44	32	0xAF0	0xFFFC0AF0	32-bit
SIUL2	IMCR47	32	0xAFC	0xFFFC0AFC	32-bit
SIUL2	IMCR50	32	0xB08	0xFFFC0B08	32-bit
SIUL2	IMCR59	32	0xB2C	0xFFFC0B2C	32-bit
SIUL2	IMCR60	32	0xB30	0xFFFC0B30	32-bit
SIUL2	IMCR61	32	0xB34	0xFFFC0B34	32-bit
SIUL2	IMCR62	32	0xB38	0xFFFC0B38	32-bit
SIUL2	IMCR63	32	0xB3C	0xFFFC0B3C	32-bit
SIUL2	IMCR64	32	0xB40	0xFFFC0B40	32-bit
SIUL2	IMCR83	32	0xB8C	0xFFFC0B8C	32-bit
SIUL2	IMCR84	32	0xB90	0xFFFC0B90	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
SIUL2	IMCR85	32	0xB94	0xFFFC0B94	32-bit
SIUL2	IMCR86	32	0xB98	0xFFFC0B98	32-bit
SIUL2	IMCR87	32	0xB9C	0xFFFC0B9C	32-bit
SIUL2	IMCR100	32	0xBD0	0xFFFC0BD0	32-bit
SIUL2	IMCR101	32	0xBD4	0xFFFC0BD4	32-bit
SIUL2	IMCR102	32	0xBD8	0xFFFC0BD8	32-bit
SIUL2	IMCR103	32	0xBDC	0xFFFC0BDC	32-bit
SIUL2	IMCR105	32	0xBE4	0xFFFC0BE4	32-bit
SIUL2	IMCR106	32	0xBE8	0xFFFC0BE8	32-bit
SIUL2	IMCR109	32	0xBF4	0xFFFC0BF4	32-bit
SIUL2	IMCR110	32	0xBF8	0xFFFC0BF8	32-bit
SIUL2	IMCR112	32	0xC00	0xFFFC0C00	32-bit
SIUL2	IMCR113	32	0xC04	0xFFFC0C04	32-bit
SIUL2	IMCR136	32	0xC60	0xFFFC0C60	32-bit
SIUL2	IMCR137	32	0xC64	0xFFFC0C64	32-bit
SIUL2	IMCR165	32	0xCD4	0xFFFC0CD4	32-bit
SIUL2	IMCR166	32	0xCD8	0xFFFC0CD8	32-bit
SIUL2	IMCR169	32	0xCE4	0xFFFC0CE4	32-bit
SIUL2	IMCR171	32	0xCEC	0xFFFC0CEC	32-bit
SIUL2	IMCR172	32	0xCF0	0xFFFC0CF0	32-bit
SIUL2	IMCR173	32	0xCF4	0xFFFC0CF4	32-bit
SIUL2	IMCR174	32	0xCF8	0xFFFC0CF8	32-bit
SIUL2	IMCR175	32	0xCFC	0xFFFC0CFC	32-bit
SIUL2	IMCR176	32	0xD00	0xFFFC0D00	32-bit
SIUL2	IMCR177	32	0xD04	0xFFFC0D04	32-bit
SIUL2	IMCR178	32	0xD08	0xFFFC0D08	32-bit
SIUL2	IMCR179	32	0xD0C	0xFFFC0D0C	32-bit
SIUL2	IMCR180	32	0xD10	0xFFFC0D10	32-bit
SIUL2	IMCR181	32	0xD14	0xFFFC0D14	32-bit
SIUL2	IMCR182	32	0xD18	0xFFFC0D18	32-bit
SIUL2	IMCR183	32	0xD1C	0xFFFC0D1C	32-bit
SIUL2	IMCR184	32	0xD20	0xFFFC0D20	32-bit
SIUL2	IMCR185	32	0xD24	0xFFFC0D24	32-bit
SIUL2	IMCR186	32	0xD28	0xFFFC0D28	32-bit
SIUL2	IMCR187	32	0xD2C	0xFFFC0D2C	32-bit
SIUL2	IMCR188	32	0xD30	0xFFFC0D30	32-bit
SIUL2	IMCR189	32	0xD34	0xFFFC0D34	32-bit
SIUL2	IMCR190	32	0xD38	0xFFFC0D38	32-bit
SIUL2	IMCR191	32	0xD3C	0xFFFC0D3C	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
SIUL2	IMCR192	32	0xD40	0xFFFC0D40	32-bit
SIUL2	IMCR193	32	0xD44	0xFFFC0D44	32-bit
SIUL2	IMCR194	32	0xD48	0xFFFC0D48	32-bit
SIUL2	IMCR195	32	0xD4C	0xFFFC0D4C	32-bit
SIUL2	IMCR196	32	0xD50	0xFFFC0D50	32-bit
SIUL2	IMCR197	32	0xD54	0xFFFC0D54	32-bit
SIUL2	IMCR198	32	0xD58	0xFFFC0D58	32-bit
SIUL2	IMCR199	32	0xD5C	0xFFFC0D5C	32-bit
SIUL2	IMCR200	32	0xD60	0xFFFC0D60	32-bit
SIUL2	IMCR201	32	0xD64	0xFFFC0D64	32-bit
SIUL2	IMCR202	32	0xD68	0xFFFC0D68	32-bit
SIUL2	IMCR203	32	0xD6C	0xFFFC0D6C	32-bit
SIUL2	IMCR204	32	0xD70	0xFFFC0D70	32-bit
SIUL2	IMCR205	32	0xD74	0xFFFC0D74	32-bit
SIUL2	IMCR206	32	0xD78	0xFFFC0D78	32-bit
SIUL2	IMCR213	32	0xD94	0xFFFC0D94	32-bit
SIUL2	IMCR214	32	0xD98	0xFFFC0D98	32-bit
SSCM	ERROR	16	0x6	0xFFFF8006	16-bit
SSCM	DEBUGPORT	16	0x8	0xFFFF8008	16-bit
STCU	RUNSW	32	0x4	0xFFF44004	32-bit
STCU	CFG	32	0xC	0xFFF4400C	32-bit
STCU	WDG	32	0x14	0xFFF44014	32-bit
STCU	LBRMSW	32	0x3c	0xFFF4403C	32-bit
STCU	LB0_CTRL	32	0x100	0xFFF44100	32-bit
STCU	LB0_PCS	32	0x104	0xFFF44104	32-bit
STCU	LB1_CTRL	32	0x140	0xFFF44140	32-bit
STCU	LB1_PCS	32	0x144	0xFFF44144	32-bit
STCU	LB2_CTRL	32	0x180	0xFFF44180	32-bit
STCU	LB2_PCS	32	0x184	0xFFF44184	32-bit
STCU	LB3_CTRL	32	0x1c0	0xFFF441C0	32-bit
STCU	LB3_PCS	32	0x1c4	0xFFF441C4	32-bit
STCU	MB0_CTRL	32	0x600	0xFFF44600	32-bit
STCU	MB1_CTRL	32	0x604	0xFFF44604	32-bit
STCU	MB2_CTRL	32	0x608	0xFFF44608	32-bit
STCU	MB3_CTRL	32	0x60C	0xFFF4460C	32-bit
STCU	MB4_CTRL	32	0x610	0xFFF44610	32-bit
STCU	MB5_CTRL	32	0x614	0xFFF44614	32-bit
STCU	MB6_CTRL	32	0x618	0xFFF44618	32-bit
STCU	MB7_CTRL	32	0x61C	0xFFF4461C	32-bit

Table continues on the next page...

Table A-1. Protected registers (continued)

Module	Register	Size	Offset	Address	Protect size
STCU	MB8_CTRL	32	0x620	0xFFF44620	32-bit
STCU	MB9_CTRL	32	0x624	0xFFF44624	32-bit
STCU	MB10_CTRL	32	0x628	0xFFF44628	32-bit
STCU	MB11_CTRL	32	0x62C	0xFFF4462C	32-bit
STCU	MB12_CTRL	32	0x630	0xFFF44630	32-bit
STCU	MB13_CTRL	32	0x634	0xFFF44634	32-bit
STCU	MB14_CTRL	32	0x638	0xFFF44638	32-bit
STCU	MB15_CTRL	32	0x63C	0xFFF4463C	32-bit
STCU	MB16_CTRL	32	0x640	0xFFF44640	32-bit
STCU	MB17_CTRL	32	0x644	0xFFF44644	32-bit
STCU	MB18_CTRL	32	0x648	0xFFF44648	32-bit
STCU	MB19_CTRL	32	0x64C	0xFFF4464C	32-bit
STCU	MB20_CTRL	32	0x650	0xFFF44650	32-bit
STCU	MB21_CTRL	32	0x654	0xFFF44654	32-bit
STCU	MB22_CTRL	32	0x658	0xFFF44658	32-bit
STCU	MB23_CTRL	32	0x65C	0xFFF4465C	32-bit
STCU	MB24_CTRL	32	0x660	0xFFF44660	32-bit
STCU	MB25_CTRL	32	0x664	0xFFF44664	32-bit
STCU	MB26_CTRL	32	0x668	0xFFF44668	32-bit
WKPU	NCR	32	0x8	0xFFF98008	32-bit
XOSC	CTL	32	0x0	0xFFFB0080	32-bit

Appendix B
Release Notes

B.1 General changes throughout and Preface

- System Status and Configuration Module (SSCM)
  - For BMODE of SSCM System Status (SSCM_STATUS) added description of setting 010b for SCI Serial Boot Loader

- LINFlexD :
  - In UART mode features, added a bullet "12-bit + parity reception"
  - In UART mode, added "13-bit frames" in existing list
  - In Introduction, Updated paragraph as "The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support."
  - Added 13-bit frames as a new section
  - In RDFL_RFC, Updated description at 3b'010 and 3b'100
  - In UART Mode Control Register (LINFlexD_UARTCR), added WLS bit and removed Reserved bit
  - In UART receiver, added two notes starting with "When WLS bit = 1...."

- Mode Entry Module (MC_ME) :
  - MC_ME_PS3[S_FLEXRAY] updated as Read-only bit.

- Analog-to-Digital Converter (ADC) :
  - In Main Configuration Register (ADC_MCR), Added ADC_MCR[TRGEN] and ADC_MCR[EDGE] fields. Earlier these fields were marked as Reserved.

- ADC Configuration :
  - Added a Note in DMA interface section

B.2 Introduction changes

- In Table 2-1, ADC Analog Reference voltage updated as “3.15 V to 5.5 V”.

B.3 Embedded Memory changes

- No substantial content changes

B.4 Signal Description changes

- In Table 4-7 short signal description of:
  - Port pin I[5] is changed from "LFAST PLL Phase 0 clock on negative terminal" to "SIPI/LFAST PLL Phase 0 clock on negative terminal"
  - Port pin C[12] is changed from "LFAST PLL Phase 0 clock on positive terminal" to "SIPI/LFAST PLL Phase 0 clock on positive terminal"
- In LVDS pins/balls deleted ‘Debug LFAST’ from footnote in table.

B.5 Memory Map changes

- In Table 5-1, added footnote in DMAMUX_0 and DMAMUX_1 entries at PCTL 36 and PCTL 146 respectively

- In Table 5-6, In DCF Records row :
  - Updated End address column to 0x004009FF
  - Updated Allocated size (bytes) column to 2048

- In Table 5-1, the Reserved row from 0xFFC08000 to 0xFFC0C003 address is now split into two separate rows:
  - 0xFFC08000 to 0xFFC0BFFF - Reserved
  - 0xFFC0C000 to 0xFFC0C003 - Generic Control Register

B.6 Functional Safety changes

- No substantial content changes

B.7 Chip Configuration changes

- Rewrote FCCU chip-specific register reset values, formerly titled "FCCU register default values." Added rows for the FCCU_CFG_TO and FCCU_NCF_TO registers. Because the information about the reset value of FCCU_CFG after a configuration timeout is now covered in the register description, deleted the related information in the row for the FCCU_CFG register. Corrected the reset value of FCCU_NCFS_CFG0 from 0002_0000h to 0000_0000h.

- Reset value of SSCM_MEMCONFIG[MREV] is changed form 1h to 2h in SSCM_MEMCONFIG reset value section.

- Added Motor control configurations as a new section
- In Table 7-33
  - Added footnote to NCF[14]
  - Previous errata e7858 integrated into reference manual, added footnote to NCF[26]
  - Updated the signal description for NCF[33]
  - Removed "Default Functional Reset (Short, Long, None)" column from this table and added a note: "Functional reset is not generated by any of the fault by default (after reset)."
  - Updated Signal description column for various NCF's.
- In Table : Corresponding error sources for OFLWn registers, added footnote in “Core I-Cache - MEMU_SYS_RAM_OFLW0: bit 1” and “Core D-Cache - MEMU_SYS_RAM_OFLW0: bit 2” rows.

- In Platform Configuration Module (PCM) :
  - Reset value of PCM_IAHB_BE1 register changed to 0007_0707h.

• Reset value of PCM_IAHB_BE2 register changed to 0007_0707h. • In PCM_IAHB_BE1 register : Access for the reserved bit fields [5:7], [21:23] and [29:31] are changed to R/W. • In PCM_IAHB_BE2 register : Access for the reserved bit fields [13:15] and [29:31] are changed to R/W. • PCM_IAHB_BE1[PRE] changed to Reserved bit. • PCM_IAHB_BE2[PRE] changed to Reserved bit. • Added Note in Periodic Interrupt Timer (PIT) configuration section
• In System Status and Configuration Module (SSCM) configuration added a note "UART and SCI are used interchangeably in the document beyond this point."
• In Table 7-16, removed the redundant Reserved rows from the table for better readability • In Table 7-12, updated OPACR value for SAR ADC_0 to 127
• In Ethernet (ENET) configuration : • Previous errata e7251 integrated into reference manual : added a Note • Added a new paragraph : "The ENET module's RMII_CLK can be output ...." • Added a new sub section : Generic Control Register
• Added a footnote in Figure 7-8
• Added a Note in Flash Memory Controller (PFLASH) configuration section
• In Debug section, added Table 7-34
• Updated Software Watchdog Timer (SWT) configuration section
• Updated Table 7-3
• Updated paragraph text in Fault inputs section • Previous errata e8229 integrated into reference manual : added a new section FCCU chip-specific glitch-filter guidelines

B.8 Reset changes

• No substantial content changes

B.9 DCF Records changes

• In DCF clients available in the SOC section : Updated table

B.10 Device Security changes

• No substantial content changes

B.11 Debug changes

• Previous errata e8014 integrated into reference manual, added a note in Introduction section.
• In DID register reset and parameter values, added a new maskset bullet point as: Maskset N15P: 29B4501Dh

- Added a new section Device identification register reset values.
- Previous errata e6726 integrated into reference manual : In Features, added Note - "MCKO may be gated one clock period early when ...."

B.12 Power Management changes

- In Power-on Reset the slow ramp up / ramp down of battery supply and following VDD is changed from 0.5 V / min. to 0.9 V / sec.

B.13 Clocking changes

- Updated Figure 13-1, to show PLL1 as a source for AUX Clock Selector 0. - Updated Figure 13-3 :BIU clock input signal connection changed from MOTC_CLK to PGBRIDGEx_CLK for CTU and SGEN blocks.
- Previous errata e8227 integrated into reference manual, added note in Peripheral clocks section. - Updated paragraph text in Clock monitoring - In Clock generation, added a Note.

B.14 e200z4 Core Complex Overview changes

- No substantial content changes

B.15 Core description module changes

- Table 15-4 :
  - Correction: se_lbz latency is 2 (was previously stated as 22)
  - Clarification: Footnote added to se_lwz latency ("Aligned")
  - Clarification: Footnote added to se_sth latency ("Aligned")
  - Clarification: Footnote added to se_stw latency ("Aligned")
- Table 15-5 :
  - Clarification: Footnote added to divwo[.] latency ("With early-out capability, timing is data-dependent.")
  - Clarification: Footnote added to divwu[.] latency ("With early-out capability, timing is data-dependent.")
  - Clarification: Footnote added to divwuo[.] latency ("With early-out capability, timing is data-dependent.")
  - Clarification: Footnote added to e_lhau latency ("Aligned.")
  - Clarification: Footnote added to lhaux latency ("Aligned.")
  - Clarification: Footnote added to lhax latency ("Aligned.")
  - Clarification: Footnote added to lhbrx latency ("Aligned.")
  - Clarification: Footnote added to e_lhz latency ("Aligned.")
  - Clarification: Footnote added to e_lhzu latency ("Aligned.")
  - Clarification: Footnote added to lhzux latency ("Aligned.")
  - Clarification: Footnote added to lhzx latency ("Aligned.")
  - Clarification: Footnote added to lwbrx latency ("Aligned.")
  - Clarification: Footnote added to e_lwz latency ("Aligned.")

  - Clarification: Footnote added to e_lwzu latency ("Aligned.")
  - Clarification: Footnote added to lwzux latency ("Aligned.")
  - Clarification: Footnote added to lwzx latency ("Aligned.")
  - Clarification: Footnote added to mfdcr latency ("Plus additional synchronization time.")
  - Clarification: Footnote added to mfspr (DEBUG, CACHE, LMEM < MPU) latency ("Plus additional synchronization time.")
  - Clarification: Footnote added to mpure latency ("Plus additional synchronization time.")
  - Clarification: Footnote added to mpuwe latency ("Plus additional synchronization time.")
  - Clarification: Footnote added to msync latency ("Plus additional synchronization time.")
  - Clarification: Footnote added to mtmsr latency ("Plus additional synchronization time.")
  - Clarification: Footnote added to mtspr (DEBUG, CACHE, LMEM < MPU) latency ("Plus additional synchronization time.")
  - Clarification: Footnote added to e_sth latency ("Aligned.")
  - Clarification: Footnote added to sthbrx latency ("Aligned.")
  - Clarification: Footnote added to e_sthu latency ("Aligned.")
  - Clarification: Footnote added to sthux latency ("Aligned.")
  - Clarification: Footnote added to sthx latency ("Aligned.")
  - Clarification: Footnote added to e_stw latency ("Aligned.")
  - Clarification: Footnote added to stwbrx latency ("Aligned.")
  - Clarification: Footnote added to e_stwu latency ("Aligned.")
  - Clarification: Footnote added to stwux latency ("Aligned.")
  - Clarification: Footnote added to stwx latency ("Aligned.")
- Table 15-6 :
  - Clarification: Footnote added to efsmsub throughput ("Destination register is also a source register, so for full throughput, back-to-back operations must use a different destination register.")
  - Clarification: Footnote added to efsnmadd throughput ("Destination register is also a source register, so for full throughput, back-to-back operations must use a different destination register.")
  - Clarification: Footnote added to efsnmsub throughput ("Destination register is also a source register, so for full throughput, back-to-back operations must use a different destination register.")
- Table 15-8 :
  - Clarification: Footnote added to throughput ("Timing is data-dependent.")
- Table 15-18 :
  - For Reserved bits 8:11, 13:14, 19:24, and 28, changed footnote number from 2 to 1

- Instruction timing :
  - Changes to Instruction class cycle counts (Table 15-3):
    - "Description" column heading changed to "Throughput"
    - "Integer: compare" latency is 1 (previously was no value)
    - "Integer: compare" throughput is 1 (previously was no value)
    - Extraneous row (Branch with Latency 3/2/1) deleted
  - Changes to Instruction timing by mnemonic—16-bit instructions (Table 15-4):
    - Correction: Previously, se_cmpl instruction appeared twice. Second instance changed to se_cmpli
  - Changes to Instruction timing by mnemonic—32-bit instructions (Table 15-5):
    - Correction: Extraneous ">" suffix deleted from addco[.] instruction
    - Correction: e_sc latency value is 4 (previously stated as 1).
    - Correction: Latency value for mtspr (except DEBUG, msr, hid0/1) is 1 (previously stated as "i").

- Machine State Register (MSR) :
  - Bits added to register figure and field descriptions table:
    - SPV (SP/Embedded FP/Vector available)
    - WE (Wait State (Power management) enable)
    - FP (Floating-Point Available)
    - FE0 (Floating-point exception mode 0)
    - FE1 (Floating-point exception mode 1)
    - IS (Instruction Address Space)
    - DS (Data Address Space)

NOTE: None of the above bits are functional. They were previously shown as reserved bits but are redefined to clearly show any deviations from Book III-E of the Power ISA version 2.06 specification.

- Dual-issue operation :

• Footnote added to items in Concurrent instruction issue capabilities table: "Excludes divide/sqrt class instructions occurring in both issue slots" • Instruction timing : • Minor editorial (non-technical) changes • Changes to Instruction class cycle counts table (Table 15-3): • Throughput value for load multiple and store multiple instruction classes is 1 + *n*/2
• DMEM Control Register 0 (DMEMCTL0) : • First sentence of DSWCE field value 11 description changed to, "Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte) writes." (was "...(1-, 2-, or 3-byte) writes.")
• Branch Unit Control and Status Register (BUCSR) : • New section
• Local Data memory overview : • Content added describing front door and backdoor accesses to local Data memory • Editorial changes
• Local Data memory overview : • Clarification: "The local memory is dual-ported, with a dedicated port for CPU accesses and a shared slave port for external accesses" changed to "The local memory has one dedicated port for CPU accesses and another shared slave port for external accesses"

B.16 SIUL2 module changes

• Added note in the bit description for ODE bit under SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR*n*) register. • Editorial change made to SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR) register description
• Previous errata e7788 integrated into the RM: Added text to the existing note ("unless it is 8 bit.....reserved register space") to "Memory map and register description" section.

B.17 Crossbar switch module changes

• Changed occurrences of "AXBS" or "AXBS module" to "Crossbar Switch."
• General operation : Removed phrase, ", other than the flash (if present),". Removed last paragraph that began with "If present, the flash slave port..."
• Features : • Replaced "64-bit data bus" with "Up to single-clock 64-bit transfer". • Replaced "Support for byte, 2-byte, 4-byte, 8-byte, and 32-byte burst transfers" with "Support for burst transfers of up to 16 beats of data". • Memory Map / Register Definition :Edited introductory text and note. Added paragraph to note beginning with , "All references to the crossbar switch..." • Added content to note in Control Register (XBAR_CRS*n*) description about HPE fields.
• Removed bullet beginning with, "Operation at a 1-to-1 clock frequency..." from Features.

B.18 XBIC module changes

• Overview : Removed content which is not useful to customer.

- Features : Removed bullet beginning with "hready, hresp0, and hresp2..."
- Functional description : Section has been rewritten. Removed "XBIC H Matrix Definition" table.
- Block diagram :
  - Replaced cross references to figures with "the following diagram."
  - In XBIC system block diagram, corrected label on top green signal to "hready, hresp0, hresp2".
  - Removed figure, "XBIC block diagram".

- Changed "last transfer" to "first transfer" in XBIC_EAR register description.

- Minor editing.
- XBIC Error Status Register (XBIC_ESR) : Previous errata e8310 integrated into the RM: Added note to register description.
- XBIC Error Address Register (XBIC_EAR) : Previous errata e8310 integrated into the RM: Added note to register description.

B.19 AIPS module changes

- Edited General operation.

B.20 SMPU module changes

- In Figure 20-1 of Block diagram section: in upper right corner, changed "Internal peripheral bus" to "Register access"
- In Features section:
  - Removed what was the first list item: "Support for a maximum of 16 program-visible 128-bit region descriptors, accessible as four 32-bit words each. "
  - Changed first (formerly second) list item from "Each region descriptor defines an arbitrarily-sized space, aligned anywhere in memory" to "Arbitrarily sized protection regions alignable anywhere in memory."
  - Moved first five list items to highest level in list hierarchy
- For Error Address Register, Bus Master n (SMPU_EAR*n*) changed reset value from "Undefined at reset" to 0000_0000h
- For Error Detail Register, Bus Master n (SMPU_EDR*n*) changed reset value from "Undefined at reset" to 0000_0080h
- In Hit determination section:
  - Replaced Boolean equation for hit determination with new sentence: "When RGD*n*_Word3[VLD] is 1, a region hit occurs when the requested address is between the start address (RGD*n*_Word0[SRTADDR]) and the end address (RGD*n*_Word1[ENDADDR]), inclusive."
  - Changed first Note from "The SMPU does not verify that ENDADDR >= SRTADDR" to "If ENDADDR < SRTADDR, no hit occurs."
- Changed title of Final evaluation and error terminations section (formerly was "Putting it all together and error terminations")

B.21 INTC module changes

- In Interrupt sources, changed "chapter that describes how modules are configured and connected" to "chip-specific INTC information".

- In TIMERn, changed INUM to IRQ.

- In Block diagram, revised the stem sentence and figure title to indicate that the diagram is an example of a four-processor configuration, and that the actual number of supported processors is given in the chip-specific INTC information.

• In Features, ensured that the bulleted list shows only those processors supported on this chip. • Revised Memory map and register definition to show only those registers and fields actually present on this chip.
• In Features, changed "monitor the latency of up to 4 interrupt sources per corresponding Processor" to "monitor the latency of interrupt sources for a corresponding Processor". • In Memory map and register definition, changed "Although INTC_SSIn are 8 bits wide..." to "Although INTC_SSCIRn are 8 bits wide...".
• In Memory map and register definition : • Added Figure 21-2 and surrounding text explaining write restrictions for PSR*n* and SSCIR*n*. • In INTC Priority Select Register (INTC_PSR*n*) and INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*), added "See Figure 21-2 for limitations on writing to this register."

B.22 eDMA module changes

• Changed description for value b101 of TCD_ATTR[SSIZE] to "32-byte burst (4 beats of 64 bits)". • eDMA initialization : Changed "ERQ register" to "ERQH and ERQL registers" in bullet 5.
• Made several editorial corrections and improvements. • Made editorial change to Features : "data packet" to "transferred data".
• Added note to DMA_DCHMIDn[EMI] bit field description.
• Fault reporting and handling : Added note re. cancel transfer request. Added note re. channel priority errors. • Block parts : Changed "16 bytes of register storage" to "a data buffer" in Data path description. • Added note to DMA_CR[CLM] description re. restriction on use of continuous link mode.
• In section "Peak transfer rates", added a note stating "All architectures will not meet the assumptions listed above. See the SRAM configuration section for more information."
• Features : Removed bullet beginning with, "Support to cancel transfers..." • Made editorial changes in Fault reporting and handling.
• Editorial changes.
• Removed "(INTC)" from Interrupt Request Register (DMA_INT) description. • Changed the following descriptive names of registers and fields: • TCD Minor Byte Count (Minor Loop Disabled) to TCD Minor Byte Count (Minor Loop Mapping Disabled). • TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) to TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled). • TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) to TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled). • Link Channel Number to Minor Loop Link Channel Number (DMA_TCDn_CITER_ELINKYES[LINKCH]). • Link Channel Number to Major Loop Link Channel Number (DMA_TCDn_CSR[MAJORLINKCH]).
• Edited Introduction. • Edited note in UCE field description in Error Status Register (DMA_ES). • Dynamic channel linking : Added cross-reference to TCD structure.
• Edited Features.
• Edited CLM description.
• Error Status Register (DMA_ES) : Added two causes of channel errors to list in register description.

B.23 DMAMUX module changes

• Removed the address information of CHCFG1 register (base address + 0x01) and CHCFG8 (base address + 0x08) from the section "Enabling and configuring sources."

B.24 EIM module changes

- Edited throughout
- In Memory map and register definition introduction text, changed register mnemonic from "Word" to "WORD"
- Error Injection Channel Descriptor, Word1 (EIM_EICHD*n*_WORD1)
    - Changed name of field from Upper Data Mask (UDATA_MASK) to Data Mask Bytes 0-3 (B0_3DATA_MASK), and changed references to the field accordingly
    - In B0_3DATA_MASK description, added text: "For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 0-3 of the read data bus, see the chip-specific EIM information."
- Error Injection Channel Descriptor, Word2 (EIM_EICHD*n*_WORD2)
    - Changed name of field from Lower Data Mask (LDATA_MASK) to Data Mask Bytes 4-7 (B4_7DATA_MASK), and changed references to the field accordingly
    - In B4_7DATA_MASK description, added text: "For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 4-7 of the read data bus, see the chip-specific EIM information."
- Functional description
    - Added a Note: "When the use case for a channel requires writing any EICHD_WORD register, write the EICHD_WORD register before executing the two-stage enable mechanism. A successful write to any EICHD_WORD register clears the corresponding EICHEN[EICHEN] field."

- Overview
    - Added Note preceding diagram: "The following diagram shows an example EIM implementation with a 64-bit read data bus and an 8-bit checkbit bus."
    - To title of Figure 24-1 added: "(64-bit read data bus and 8-bit checkbit bus)"
- Memory map and register definition
    - Reorganized first paragraph into subsection titled "Programming model access"
    - Completely revised second paragraph and reorganized it into subsection titled "Error injection channel descriptor: function and structure"
    - For Error Injection Channel Enable register (EIM_EICHEN) edited and reorganized register and field descriptions and renamed field from EICHEN to EICH0EN
    - Renamed registers and changed references to them accordingly:
        - EICHD_WORD0 to EICHD0_WORD0
        - EICHD_WORD1 to EICHD0_WORD1
        - EICHD_WORD2 to EICHD0_WORD2
    - For CHKBIT_MASK in Error Injection Channel Descriptor, Word0 (EIM_EICHD*n*_WORD0) added text to field description: "The width of the field for the channel is defined in the chip-specific EIM information."
    - Incorporated editorial changes in descriptions of Error Injection Channel Descriptor, Word1 (EIM_EICHD*n*_WORD1) and Error Injection Channel Descriptor, Word2 (EIM_EICHD*n*_WORD2)
    - Revised descriptions of B0_3DATA_MASK field in Error Injection Channel Descriptor, Word1 (EIM_EICHD*n*_WORD1) and B4_7DATA_MASK field in Error Injection Channel Descriptor, Word2 (EIM_EICHD*n*_WORD2)
- Edited all content in Functional description

- In Memory map and register definition changed first sentence:
    - from: "The EIM provides an IPS programming model mapped to a standard 16 KB slot for an on-platform peripheral."
    - to: "The EIM provides an IPS programming model mapped to an on-platform peripheral slot."

- Memory map and register definition
    - Minor editorial changes: in each list item of first two unordered lists, capitalized first letter of first word

---

- Memory map and register definition
    - Edited and reorganized text in Error injection channel descriptor: function and structure

---

B.25 PLLDIG changes

• Editorial updates throughout. • Clock configuration   • Updated equations in section.
• PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)   • Updated 'PLL0' to 'PLL1' in Register description, "...effective after PLL1 is disabled, then reenabled."
• PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)   • Replaced the sentence, "The values of PREDIV and MFD..." with "PLL0DV can be modified at anytime..." in the register description. • PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)   • Updated register description. • PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)   • Removed the sentence, "Changing the values of PLL1FM[MODEN] and PLL1FM[MODSEL] fields..." in the register description. • Clock configuration   • Add equation headings and titles to equations.
• Removed section "Maximum Lock Time".
• PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)   • Removed the text ", however, the output clock from PLL..." from the last sentence in the register description. • Normal mode with reference, PLL0 or both PLLs enabled   • Replaced "output divider (Reduced Frequency Divider) and whether PLL modulating is enabled" with "and modulation enable are" in second paragraph.   • Updated the end of the first sentence, "In normal mode, PLL0 receives..." • Introduction   • Removed "The chip provides a user interface and control over" from the first sentence which now reads "The Dual PLL system composed of..."
• PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)   • Added updated content to the register description. • PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)   • Added updated content to the register description.
• Removed the section "Acronyms and Abbreviations". • Frequency modulation   • Updated the equation in the Note, from "$f_{max} = f_{sys} \times (1 + (MD\%)).$" to "$f_{max} = f_{sys} \times (1 + (MD\% / 100)).$"
• PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)   • Removed the text "the binary equivalent of" from the field descriptions of INCSTP and MODPRD.
• PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)   • Updated reset value of reserved field at location 21(Reserved), and changed the access to RO.
• Frequency modulation   • Updated the sentence "If center modulation is selected..." to "If center modulation is used..." in the Note "The device maximum operating frequency..."
• Register PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)   • Replaced the register description with updated content.
• PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)   • Updated MODSEL bitfield value descriptions for both 0 and 1.
• PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)   • Removed the sentence "Modulation period (PLL1FM[MODPRD]) and increment step..." from the register description. • PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)   • Removed the sentence "The dither disable (PLLDIG_PLL1FD[DTHDIS]) and..." from the register description.
• Frequency modulation

  - Updated figure Figure 25-1 to show correct modulation depth.

- In PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)
  - Replaced DTHDIS field with Reserved field at offset 15.

B.26 CMU changes

- CLKMN1 supervisor
  - Updated NOTE to read "(HFREF$_{Actual}$ = (HFREF$_{Ideal}$ ÷ 0.95)" instead of "(HFREF$_{Actual}$ = (HFREF$_{Ideal}$ + 2) × 1.05)".
  - Added "The actual LFREF value will be 762 when the accuracy is taken into consideration (LFREF$_{Actual}$ = (LFREF$_{Ideal}$ ÷ 1.05)." to the NOTE.

- Main features
  - Updated second bullet to "CLKMN0_RMT monitoring with respect to CLKMT0_RMN ÷ $2^{CSR[RCDIV]}$ clock."
- Added CMU_ prefix in front of register/field names that did not previously include it.
- CMU Frequency Display Register (CMU_FDR)
  - Updated register description placing items in a bullet list.
- CMU Interrupt Status Register (CMU_ISR)
  - Updated Note in register description, "All flags in the CMU_ISR are set..."
  - Updated bitfield description FHHI
  - Updated bitfield description FLLI
- Frequency meter
  - Added "[FD]" to end of CMU_FDR throughout where it was not previously included.

- CMU Interrupt Status Register (CMU_ISR)
  - Updated the Note in the OLRI bitfield description, "When attempting to enter STOP mode..."

B.27 MC_CGM changes

- Previous errata e7103 integrated into reference manual, added a note "If the current system clock source.....of the switch." in SELSTAT bit of System Clock Select Status Register (MC_CGM_SC_SS) register.
- Previous errata e7250 integrated into reference manual, added a note "Always write.....registers." and changed bit access from RO to RW in DE bit of System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0) register.

- Updated the bit-width of MC_CGM_AC0_SC[SELCTL] to [5:7] and added the clock source availability for PLL1 PHI at 100b.
- Updated the bit-width of MC_CGM_AC0_SS[SELSTAT] to [5:7] and added the clock source availability for PLL1 PHI at 100b.
- Added Note in MC_CGM_AC0_SC register description.

B.28 XOSC changes

- No substantial content changes

- XOSC Control Register (XOSC_CTL)
  - Updated XOSC_CTL[I_OSC] field access.

B.29 IRCOSC changes

- Frequency trimming calculation
  - Updated table contents replacing δF_{var_SW} with δf_{TRIM}.
  - Updated all instaces of "F" with "*f*" to represent frequency.

B.30 PRAMC module changes

- SRAM controller memory map and register definition :
  - Note added regarding caution required when reconfiguring platform RAM controller during device operation
- e2eECC considerations on less-than-64-bit write transactions :
  - Added to end of section: "Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The RAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module."
- Less than 64-bit writes :
  - Added single bit and multi bit error signals to Read-modify-write data path figure
  - Changed hardware signal names to names that describe signal function more clearly
- Optional read wait-state :
  - All references to the PRCR1 register changed to "PRCRx". This is only an editorial change--there is no change to any register name or function.
- Read and write operations (formerly "Read / Write introduction" :
  - Section completely rewritten

B.31 Flash memory controller module changes

- Functional description :
  - Deleted from first sentence: "As shown in the block diagram"
- ECC on data flash accesses :
  - Editorial changes
  - Clarification: Following note added: "EEPROM should be avoided for storage of executable code."
- Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE) :
  - Clarification: Only CRD0EN - CRD7EN fields (bits 0 - 7) are valid for this device because it can have a maximum of 8 overlay regions. The remaining bits are read only zero reserved bits.
- Platform Flash Remap Control Register (PFLASH_PFCRCR) :
  - Correction: PFLASH_PFCRCR[SAFE_CAL] field description updated to state the correct reduced number of calibration regions available if SAFE_CAL=1. When the SAFE_CAL field is set, the number of available calibration regions is reduced by half. For this device, if SAFE_CAL=1, the number of calibration regions available is reduced to 4.
- Platform Flash Configuration Register 1 (PFLASH_PFCR1) :
  - Clarification: The PFLASH_PFCR1[P0_BFEN] field can only be updated when PFLASH_PFCR3[BFEN_LK]=0.
- Platform Flash Configuration Register 3 (PFLASH_PFCR3) :
  - Reserved field at bit 15 deleted
  - New field added at bit15: BFEN_LK. This field must be set to 0 before the PFCR1[P0_BFEN] field can be updated.

• Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRDn_Word0) : • Clarification: Previously the LSTARTADDR field was shown as a 32-bit field with read/write access. The address is a 32-bit address but the least significant 4 bits are always 0 and are now shown as a reserved read only zero field. • Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRDn_Word1) : • Clarification: Previously the PSTARTADDR field was shown as a 32-bit field with read/write access. The address is a 32-bit address but the least significant 4 bits are always 0 and are now shown as a reserved read only zero field.
• Platform Flash Configuration Register 1 (PFLASH_PFCR1) : • Clarification: All "1xx" values in the APC field have the same effect. Previously, the values, 100, 101, 110, and 111 in the APC field were listed individually and had identical descriptions. They are now condensed into a single entry, "1xx" in the value list to reduce the possibility of confusion.
• Platform Flash Configuration Register 1 (PFLASH_PFCR1) : • Note added to PFLASH_PFCR1[APC] field description: "A value of '1' in the most significant bit causes the flash controller to function in retrograde/legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup."
• ECC on data flash accesses : • Clarification: If a non-correctable error is detected, the flash memory controller returns a value of FFFF_FFFFh to the requesting master.
• Access protections : • Content previously located in the "Error termination" and "Flash error response operation" sections moved to this section. • Error termination : • Section deleted • Flash error response operation : • Section deleted • Read cycles - buffer miss : • Section deleted • Read cycles - buffer hit : • Section deleted
• Throughout: • Editorial changes
• Platform Flash Configuration Register 3 (PFLASH_PFCR3) : • Correction: Access type of bit 11 in the PFLASH_PFCR3 register has been changed from read-only to read/write. The bit is reserved and has no underlying function.
• Features : • Correction: "Nexus trace stream interface supports message dumping to the on-chip overlay RAM" removed from features list (feature is only supported in devices with dedicated on-chip overlay RAM) • Functional description : • Correction: "Nexus trace" removed from list of modules interfaced with the flash memory controller (The Nexus interface is only supported by the flash memory controller in devices with dedicated on-chip overlay RAM) • PFlash calibration remap support : • Correction: "Can be used to hold device debug trace stream" removed from list of features supported by the overlay function (This feature is implemented only in devices with dedicated on-chip overlay RAM) • Correction: "Timing for calibration accesses to a particular overlay RAM is only guaranteed if it is not being used as a destination for trace streaming at the same time" removed (Trace streaming is only available in devices with dedicated on-chip overlay RAM)
• Platform Flash Configuration Register 1 (PFLASH_PFCR1) : • Clarification: Following statement added to PFLASH_PFCR1[APC] field description: "For valid RWSC and APC combinations please refer to the device data sheet. • Following note in PFLASH_PFCR1[APC] field description reformatted into note list: "A value of '1' in the most significant bit causes the flash controller to function in retrograde/legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup." • Added to note list in PFLASH_PFCR1[APC] field description: "Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the data sheet."

- Clarification: Updated portion of PFLASH_PFCR1[RWSC] field description that points to data sheet for valid values to state, "For valid RWSC and APC combinations please refer to the device data sheet. (was "The required settings are documented in the device data sheet")
- Clarification: Following statement in PFLASH_PFCR1[RWSC] field description reformatted as item in note list for emphasis: "Higher operating frequencies require non-zero settings for this field for proper flash operation"
- Clarification: Added to note list in PFLASH_PFCR1[RWSC] field description: "Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the device data sheet"

B.32 c55fmc module changes

- Features :
  - Clarification: Feature "Triple Voted Flops for flash functions requiring high reliability" changed to, "Triple Voted Flops for flash functions requiring high reliability, e.g., internal trimming, redundancy, and mode control"

- Over-Program Protection 0 register (C55FMC_OPP0) :
  - Correction: Two most significant bits of the register always reset to 0 (previously shown as dependent on chip configuration)
  - Clarification: Footnote in register diagram replaced with footnotes stating that the reset value of each field (LOWOPP and MIDOPP) varies among devices
- Over-Program Protection 1 register (C55FMC_OPP1) :
  - Correction: Sixteen most significant bits of the register always reset to 0 (previously shown as varying among devices)
  - Clarification: Footnote in register diagram replaced with footnote stating that the reset value of the HIGHOPP field varies among devices
- Over-Program Protection 2 register (C55FMC_OPP2) :
  - Clarification: Footnote in register diagram replaced with footnote stating that the reset value of the A256KOPP field varies among devices
- Over-Program Protection 3 register (C55FMC_OPP3) :
  - Correction: Sixteen most significant bits of the register always reset to 0 (previously shown as varying among devices)
  - Clarification: Footnote in register diagram replaced with footnote stating that the reset value of the A256KOPP field varies among devices

B.33 Flash OTP changes

- No substantial content changes

B.34 DSMC module changes

- No substantial content changes

B.35 ADC Configuration changes

- In CTU interface, added Note: "CTU Clock (MC_CLK) and ADC_CLK should either be same and synchronous or CTU can also operate with ADC_CLK being an integer plus half (1.5, 2.5,3.5, ...) of MC_CLK clock."

- Added a new topic ADC channel conversion under ADC pin muxing

B.36 ADC module changes

- Features :
  - Added list of channels for each group of ADC channels.
- In Normal conversion :
  - Added list of channels for each group of ADC channels.
- In Presampling channel enable :
  - Added list of channels for each group of ADC channels.
- In Injected conversion :
  - Added list of channels for each group of ADC channels.
- In Analog watchdog setting :
  - Added list of channels for each group of ADC channels.
- ADC clock prescaler and sample time settings :
  - Added list of channels for each group of ADC channels.

- Self test (Quick-Check/Safety mode) :
  - Clarification: ADC self test step 1, "Supply self-test (analog supply)", was previously described as "Measures the analog supply for the ADC". Changed to: "Measures the analog supply voltage (VDDA) and divides the result by the bandgap voltage (result from Algorithm S step 0). The VDDA conversion result is written to ADC_STDR1[TCDATA]. The final result (after division) consists of an integer portion and a fractional portion, both of which are written to the ADC_STDR2 register."
  - Comment added regarding pre- and post-conversion scaling of the analog supply voltage that results in the least significant bits of the VDDA conversion result always being '0'.
- Self Test Data Register 1 (ADC_STDR1) :
  - Updated register description: "The ADC_STDR1 register contains the most current result data for conversions executed during ADC self test. Additionally, the register contains status bits to indicate whether result data has been read and whether existing conversion data has been overwritten by a newer result."
  - Note added: "For Algorithm S step 1, the result data contained in this register is not the final step 1 result. The step 1 data is the conversion of the analog supply (VDDA). This VDDA conversion result is divided by the Algorithm S step 0 conversion data and the result is written to the ADC_STDR2 register."
  - Detailed note added regarding pre- and post-conversion scaling of the analog supply voltage that results in the least significant bits of the VDDA conversion result always being '0'.

- Calibration, BIST Control and status Register (ADC_CALBISTREG) :
  - Default register reset value changed to C037_0670h (was 2037_06F0h)
- Conversion Timing Register 0 (ADC_CTR0) :
  - Default register reset value value changed to 0000_0014h (was 0000_0016h)
- Conversion Timing Register 1 (ADC_CTR1) :
  - Default register reset value value changed to 0000_0014h (was 0000_0016h)

- Self Test Analog Watchdog Register 0 (STAW0R) :
  - Correction: The ADC_STAW0R[THRL] register field description has been updated to state that if the analog watchdog is enabled, the STSR1[ERRx] status bit is set if STDR1[TCDATA]) < THRL. Previously stated that the input was expected in 2's complement format and that the error bit would be set if (Absolute value of STDR1[TCDATA]) > (Absolute value of THRL). The value entered is NOT interpreted as 2' complement.
  - Correction: The ADC_STAW0R[THRH] register field description has been updated to state that if the analog watchdog is enabled, the STSR1[ERRx] status bit is set if STDR1[TCDATA]) > THRH. Previously stated that the

input was expected in 2's complement format and that the error bit would be set if (Absolute value of STDR1[TCDATA]) > THRH.. The value entered is NOT interpreted as 2' complement.

- Throughout Memory Map/Register Definition :
  - Changed Vref to VREF
  - Spelling corrections
  - Formatting changes
- Self Test Analog Watchdog Register 0 (ADC_STAW0R) :
  - Register description updated
  - Clarification. Note added to ADC_STAW0R[THRH] and ADC_STAW0R[THRL] field descriptions: "The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments."
  - ADC_STAW0R[AWDE] field description updated
  - ADC_STAW0R[WDTE] field description updated
- Self Test Status Register 3 (ADC_STSR3) :
  - Correction: Equation in ADC_STSR3[DATA0] field changed to "Test channel data = VREF/VBGAP" (was "... VBGAP/VREF")
- Self Test Analog Watchdog Register 1A (ADC_STAW1AR) :
  - Register description updated
  - Clarification. Note added to ADC_STAW1AR[THRH] and ADC_STAW1AR[THRL] field descriptions: "The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments."
- Self Test Analog Watchdog Register 1B (ADC_STAW1BR) :
  - Register description updated
  - Clarification. Note added to ADC_STAW1BR[THRH] and ADC_STAW1BR[THRL] field descriptions: "The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments."
- Self Test Analog Watchdog Register 2 (ADC_STAW2R) :
  - Register description updated
  - Clarification. Note added to ADC_STAW2R[THRL] field description: "The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments."
- Self Test Analog Watchdog Register 4 (ADC_STAW4R) :
  - Register description updated
  - Clarification. Note added to ADC_STAW4R[THRH] and ADC_STAW4R[THRL] field descriptions: "The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments."
  - ADC_STAW4R[AWDE] field description updated
  - ADC_STAW4R[WDTE] field description updated
- Self Test Analog Watchdog Register 5 (ADC_STAW5R) :
  - Register description updated
  - Clarification. Note added to ADC_STAW5R[THRH] and ADC_STAW5R[THRL] field descriptions: "The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments."

- In Calibration (High Accuracy mode), removed "For example, if the ADC is calibrated with a 3.3V reference and operated with a 5V reference, it must be recalibrated".

- Editorial updates throughout chapter.
- Conversion data processing
  - Updates throughout section.

- In bitfield JECH of ISR
    - Added NOTE, "If there is no channel selected in the ADC_JCMR*n* register(s)..." to field description.
- In bitfield ECH of ISR.
    - Added NOTE, "If there is no channel selected in the ADC_JCMR*n* register(s)..." to field description.
- Offset and Gain User Register (ADC_OFSGNUSR)
    - Removed "[15 bit accurate]" from the field descriptions of OFFSET_USER and GAIN_USER.
    - Added note "A detailed description of the use of OFSGNUSR..." to register description.
- User-defined offset and gain values and its subsections
    - Significant updates throughout.
- Conversion time
    - Changed title to 'Conversion time' from 'Compare phase time'
    - Significant updates throughout.

- NSTART
    - Added Note to the MCR[NSTART] bitfield description, "This bit cannot be set (1) when PWDN = 1. This bit is always cleared when PWDN = 1."

- Register CTR0 (Conversion Timing Register 0 (ADC_CTR0))
    - Added Note "The Bandgap voltage sampling time is controlled..." to the CTR0[INPSAMP] field description (INPSAMP).

- Self Test Status Register 2 (ADC_STSR2) :
    - Correction: OVFL bit is read-only (previously was shown as R/W)

- Presampling channel enable
    - Updated "VDD_HV_ADV (Power High voltage Supply for ADC, SGEN)" to "VDD_HV_ADV/8 (Power High voltage Supply for ADC, SGEN divided by 8)" in table Table 36-1.

- In bitfield PRECONV of PSCR
    - Added "This bit has no effect on conversion of a channel..." to the field description.

- Presampling channel enable
    - Updated formatting of table Table 36-2, added table title, and added a cross reference to the table in the preceding text.
    - Updated formatting of table Table 36-1, added table title, and added a cross reference to the table in the preceding text.
    - Added cross reference in first paragraph to the PSCR register.

- Main Configuration Register (ADC_MCR)
    - Updated the field access of Reserved field at bit location 6 (Reserved) to RW.
    - Added Note to Reserved bit field 6 description (Reserved)"User must not overwrite default value of this field."

- Calibration (High Accuracy mode)
    - Updated content throughout.

- Parts of the ADC
    - Removed "TEST Interface" bidirectional i/o from Figure 36-1.

- Parts of the ADC
    - Updated 'ADC high-level block diagram' Figure 36-1 removing the text "and external" from "(to on-chip and external mux)". It now reads "(to on-chip mux)"

- Removed the text "BCTU" throughout section "Memory Map/Register Definition".

- Crosstriggering Unit interface
    - Changed the signal name 'ctu_push' to 'ctu_EOC' in Figure 36-4.
- OPMODE
    - Updated field and field values descriptions.

- Features
    - Reformatted bullet "Modes of operation".

- Self Test Analog Watchdog Register 4 (ADC_STAW4R)
    - Updated field description of THRH.
    - Updated field description of THRL.
- Self Test Analog Watchdog Register 5 (ADC_STAW5R)

• Updated field description of THRH.   • Updated field description of THRL. • Removed section "User Defined Offset and Gain values".
• Calibration, BIST Control and status Register (ADC_CALBISTREG)   • Added the Note "This setting is used only when the application..." to the NR_SMPL field description. • Self Test Analog Watchdog Register 4 (ADC_STAW4R)   • Reformatted Note in THRL field description. • Calibration, BIST Control and status Register (ADC_CALBISTREG)   • Updated reset value of OPMODE.
• Register DMA Enable register (ADC_DMAE)   • In the DCLR bitfield value description for the case when DCLR=1, added note "When DCLR is set, ADC DMA channels should only execute read accesses on ADC_CDR*n* registers."
• Register Offset and Gain User Register (ADC_OFSGNUSR)   • Removed Note "A detailed description of the..." from register description.
• Calibration, BIST Control and status Register (ADC_CALBISTREG)   • Updated the Note in the NR_SMPL field description to "The 16 samples setting is used only when the application..."
• Calibration, BIST Control and status Register (ADC_CALBISTREG)   • Updated register reset value.

B.37 Temperature Sensor changes

• In Temperature formula, removed the equations for K1, K2, K3, K4, and dependent content. • In Temperature formula, removed the "Calibration Points" figure and its introductory sentence.
• Editorial changes and rephrased text in Introduction, Functional Description and Temperature threshold detection (digital output generation).
• "Temperature Sensor" title is changed as "Temperature Sensor (TSENS)".

B.38 SGEN changes

• In Status Register (SGEN_STAT), changed access type of SGEN_STAT[SERR] from Read/write to Write one to clear.

B.39 eTimer module changes

• Removed the word "other" from the ETIMER_CH*n*_HOLD register description and HOLD field description.
• Channel Block Diagram   • Added "eTimer Channel Block Diagram" image that removed the 'Fault' input on the 'OFLAG Control' block.
• In Channel n Status Register (ETIMER_CH*n*_STS), access for bits [6:15] has been changed from Read/write to Write one to clear.
• Editorial cleanup throughout chapter.

B.40 FlexPWM module changes

• Added register access information to all register descriptions. • Made editorial improvements throughout chapter.
• Submodule n Status Register (FlexPWM_SUB0_STS) :   • Access type for following fields changed to write 1 to clear (was R/W): REF, RF, CFX1, CFX0, CMPF   • Added reserved register at address offset 0x14A
• Added point "Capture is supported only for X channels not A and B channels" in Features section

B.41 CTU module changes

• In CTU_CR section note, added CTU_ODIS to the list of bits that can be read and set by software.
• Changed registers CTU_TGSCRR, CTU_FDCR, and CTU_CNTRNGR to 16 bits wide.
• Revised CTU_EFR[ICE] field description. • In ADC commands list, revised note to clarify command list execution. • In ADC command list format, revised command sequencing description to clarify the concept of a single command list.
• Previous errata e8137 integrated into the RM: In CTU_EFR and CTU_IFR register sections, added note to clarify flag behavior at reset.
• Revised CTU_IFR register description to read "The IFR register bits are cleared . . ." (was "The EFR register bits are cleared . . .").
• Editorial changes and improvements. • In ADC command list format, revised "Command list sequencing" figure and corresponding figure description.
• In ADC results FIFOs, changed text to "The CTU FIFOs are fixed depth" (was "The CTU supports several...").
• In Error Flag Register (CTU_EFR), revised note ("The ERRCMP field reads 0...") (was "The Tn_OE fields read 0...").
• Previous errata e7338 integrated into the RM: added note ("If the system attempts to access...") to "Memory Map and Registers" section.

B.42 STM module changes

• In Functional description   • Incorporated minor edits
• For reserved fields of following registers, added text to each field description: "This field is reserved."   • STM Control Register (STM_CR)   • STM Channel Control Register (STM_CCR*n*)   • STM Channel Interrupt Register (STM_CIR*n*)

B.43 SWT module changes

• Memory Map and Registers: incorporated various edits

- In SWT_CR register
  - In section title, changed SWR to SWT
  - Edited text in section, including in footnote

- In Servicing operations incorporated minor editorial/formatting changes

- In SWT Control Register (SWT_CR) changed Reserved from read-only and always zero to read/write

- In SWT Service Register (SWT_SR) description of WSC field, changed beginning of second sentence from "If the SWT_CR[KEY] bit is set" to "If the SWT_CR[SMD] field is 01b"
- In SWT Service Key Register (SWT_SK) description of SK field, changed beginning of second sentence from "If SWT_CR[KEY] is set" to "If SWT_CR[SMD] is 01b"

- In Overview changed final three sentences:
  - From: "If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out."
  - To: "If this servicing action does not occur before the timer expires, the SWT generates an interrupt or hardware reset request. The SWT can be configured to generate a reset request or interrupt on an initial time-out. The SWT always generates a reset request on a second consecutive time-out."
- In SWT Control Register (SWT_CR) description of ITR field
  - Added a Note: "For a description of how this chip implements SWT reset requests resulting from SWT time-outs, see the chip-specific SWT information."
  - In descriptions of field values, changed "reset" to "reset request"
- In Time-out
  - Changed the description of a reset time-out result from "reset" to "reset request"
  - Edited and reorganized entire section

B.44 PIT module changes

- Added information about register access level at the end of each register description

- In PIT_MCR register, added a note to MDIS bit field -"Always write....clock generation module."

- Added note to Example configuration for chained timers

B.45 FlexCAN module changes

- Interrupts
  - In first paragraph, added references to ECC-related interrupts
  - Added new final paragraph about ECC-related interrupts
  - Edited text and standardized format of references to registers

- In Error and Status 1 register (CAN_ESR1) description, incorporated minor edits in first and third paragraphs, and changed second paragraph:
  - from: "The CPU read action clears bits 15-10. Therefore the reported error conditions (bits 15-10) are those that occurred since the last time the CPU read this register. Bits 9-3 are status bits."
  - to: "A CPU read operation clears the following fields to 0, so these fields report error conditions that occurred after the last time the CPU read this register: BIT1ERR, BIT0ERR, ACKERR, CRCERR, FRMERR, and STFERR. TXWRN, RXWRN, IDLE, TX, FLTCONF, and RX provide status information."

- In the Overview section's last sentence of the second paragraph, changed "See the chip configuration details" to "See the chip-specific FlexCAN information"

- Changed "MCU" to "chip" throughout the chapter.

- In Table 45-10 of Rx FIFO structure

- Added "Format" to first column of first row
- Changed notation of bit ranges (in parentheses) for RXIDA, RXIDB_n, and RXIDC_n

B.46 Zipwire module changes

- In Zipwire interconnections : Updated bullet from "Tx and Rx connections to the pads via the SIUL and the MSCR registers" to "Tx and Rx configuration is controlled by LFAST Control registers".
- Removed "LFAST clocking" section.

- In Architecture : Removed paragraph text "The LVDS pads are shared with other I/O and GPIO.".

B.47 SIPI module changes

- In In ARR register added a note "This register is writeable only when SIPI_MCR[INIT] = 1." Also removed sentence, "This register can be read or written anytime." from register description.

- In CCR0[WRT], removed a note "Only transfers with channel 2 can write 1 to this bit. All other channels forces SIPI_CCRn[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming." from the WRT bit field of CCR0-CCR3 registers.
- Added bit field brief description of WRT bit in CCR2 register.

- Updated bit field access for SIPI_ACR[ADCNT] and SIPI_ACR[Reserved] to read write.

B.48 LFAST module changes

- Editorial changes.

- Removed reference to 312/320 MHz and referred to as high data rate. Updated in:
  - External signals section.
  - LFAST operating data rates section.
  - Features section.
  - Auto correlation section.
  - Figure 48-6 figure.
  - LFAST Speed Control Register (LFAST_SCR) : Updated description for bit-value 1 of RDR and TDR bit.
- Editorial changes.
- Removed table from LFAST operating data rates section.

- LVRXOP bit in LFAST LVDS Control Register (LFAST_LCR) is split into 3 individual bits:
  - LVRXOP[2] changed to LVRXOP_TR.
  - LVRXOP[1] changed to Reserved.
  - LVRXOP[0] changed to LVRXOP_BR.
- Updated the access value of LVLPEN bit to Read/Write in LFAST LVDS Control Register (LFAST_LCR).

- Updated the description of [LVCKSS] bit in LFAST LVDS Control Register (LFAST_LCR).

- In Memory map and register definition, added note "Read/Write accesses to all unimplemented registers and write accesses to Read only register will return a Transfer Error.".
- Updated the register-access value of LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR) register from R/W to W1C.

B.49 SPI module changes

• Included two topics Modified SPI Transfer Format (MTFE = 1, CPHA = 0) and Modified SPI Transfer Format (MTFE = 1, CPHA = 1).
• Reduced bit width of SPI_CTARn_SLAVE [FMSZ] from 5 to 4 and updating the bit to reserved. • In RSER register, added "Always write the reset value to this field." to some of the Reserved bits. Also, updated bit field access to RW for these bits. • In PUSHR register, added note "Always write the reset value to this field." to Reserved bits. Also, updated bit field access to RW for these bits.
• In Memory Map/Register Definition section, added RXFRn to statement re. write accesses results in transfer error. • Updated bit field description for SPI_MCR[PCSIS]. • Updated bit field description for SPI_PUSHR[PCS].
• Updated section, Continuous Serial Communications Clock
• Editorial updates.
• Updated bit field description for SPI_MCR[MTFE] • Updated SPI block diagram.
• Updated SPI_MCR[MDIS] bit field description for setting default reset value to 1 instead of 0. • In SPI_CTARn[FMSZ], updated description of 'fr' to 'register interface clock frequency'.
• Updated register description for SPI_PUSHR and SPI_PUSHR_SLAVE. • Updated bit field description and width for SPI_PUSHR_SLAVE[TXDATA]
• Added note to SPI_MCR[PCSIS] bit field. • Updated bit field description for SPI_TXFRn[TXCMD_TXDATA].
• Changed "MCU" to "chip" throughout the chapter.
• Removed note from Section, SIN—Serial Input
• In Serial Peripheral Interface (), Previous errata e6904 integrated into reference manual: Added note in SPI_RXFRn register description.

B.50 FlexRay module changes

• Memory map and register definition : Editorial updates.
• Slot Status Counter Condition Register (FR_SSCCR) : In FR_SSCCR[STATUSMASK[3:0]] bit field name, removed the text [3:0].
• In Features, added "Dual Channel Support" feature.
• In Memory map and register definition , added a paragraph, "Address offset - 0x0Ah ,0xDEh-0xE0h should not be accessed by application as corresponding feature/s are not available. Therefore, transfer error will not be generated at these offsets."

B.51 SRX/SENT module changes

• In Features changed sentence from "Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift and jitter in the Transmitter and Receiver clocks per channel" to "Supports compensation for

variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel". • In Slow Serial Message Ready Status Register (SRX_SMSG_RDY) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver". • In Fast Message Ready Status Register (SRX_FMSG_RDY) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver". • In Initialization sequence, Note removed as it was taken care of via ERR007425 • In Fast Message DMA Control Register (SRX_FDMA_CTRL) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver". • In Slow Serial Message DMA Control Register (SRX_SDMA_CTRL) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver". • In Fast Message Ready Interrupt Control Register (SRX_FRDY_IE), changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver". • In Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE) changed Note to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver". • In Input programmable filter updated value of FIL_CNT for the example shown and updated the diagram • In CRC check removed Note • In Fast Message Ready Status Register (SRX_FMSG_RDY) removed a Note • In Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL) added a Note to the field CM_PRSC • In Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL) changed sentence from "The generated Receiver Clock is compensated for variations, clock drift and jitter that might occur in the Sensor Tx Clock" to "The generated Receiver Clock is compensated for variations and clock drift that might occur in the Sensor Tx Clock". • In Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS) changed SMSG_OFLW bitfield description. • In Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG) changed description of field values of bitfield SRX_CHn_CONFIG[FCRC_TYPE] and SRX_CHn_CONFIG[SCRC_TYPE]. • In Adjustment for variation in sensor (Tx) clock deleted note that starts with "At ambient ...". • In Receiver diagnostics changed bullet point from "Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or XOR-based" to "Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or Recommended.". • In Nibble value check changed intoductory sentence from "As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 ticks:" to "As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 nibble values(11.5 ticks to 27.5 ticks):". • In CRC check changed text in introductory paragraph from "XOR-based CRC implementation" to "new recommended CRC implementation". • In Pause pulse diagnostic update changed table caption and deleted FMSG_OFLW row. • In High frequency receiver clock (protocol clock) requirements modified entire content within this heading.
• In Data Control Register 1 (SRX_DATA_CTRL1), changed reset value of each 3-bit Reserved field from 001b to 000b. • In Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL), changed access of CM_PRSC field from read/write to read-only.
• In Time stamp logic changed text: • Was: "For example, if the high frequency receiver clock is 59 MHz, the required prescaler value to generate a time stamp clock of 1 µs resolution is 60"   • Now: "For example, if the high frequency receiver clock is 60 MHz, the required prescaler value is 59 to generate a time stamp clock of 1 µs resolution"
• In DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) changed field name SRX_DMA_SMSG_BIT3[ID7_4_D3_0] to SRX_DMA_SMSG_BIT3[ID7_4_ID3_0].
• Minor editorial changes.
• In Global Control Register (SRX_GBL_CTRL) changed range of values from "1 to 255" to "0 to 255" in GBL_CTRL[TSPRC] field description.
• Previous errata e8082 integrated into reference manual: In Fast messages buffers/FIFO added the note "In the case of a fast message overflow and continuous reading of register …".

B.52 LINFlexD module changes

- Editorial updates in the following sections:
    - Slave mode
    - Identifier filtering
    - Start detection and break delimiter detection in receiver
    - UART receiver
    - UART transmitter

- Previous errata e7589 integrated into reference manual: In the LINTCSR[MODE] field description added NOTEs.

- Added list item, "When IPG_STOP is requested..." in section: Use cases and limitations which was previously deleted.

- Updated the note: "When the MODE bit is 0, any activity on the transmit ..." written in the register description of LINTCSR.
- Updated the reset value note written in the IOPE bit reset value of register LINCR2.

- In section: Master mode, changed 'BDR[0:7]' to 'BDRL and BDRM'

- In section: Memory map and register description,
    - Updated the note: "In Master mode, the registers IFCR0 – IFCR15...". Added table: "Offsets of the register from offsets 8C to 9C", below the note.
    - Minor editorial updates in register 'LINFlexD_LINCR1', bit field 'LASE'
    - Removed phrase "Parity sent is Even" from the bit field description of bits 'WL0' and 'WL1' of register LINFlexD_UARTCR

- In section: Memory map and register description,
    - Removed reset value of register 'LINFlexD_LINSR' and added the following note to the bit reset value of bit field 'RDI:
        - "Reset value of RDI reflects the RX pin state"
    - Updated the bit field description of bit fields 'Reserved' and 'DTE' of register 'LINFlexD_DMATXE'.
    - Updated the bit field description of bits fields 'Reserved' and 'DRE' of register 'LINFlexD_DMARXE'.
    - Removed description/note related to 'ROSE' bit from registers: UARTSR and LINFBRR respectively.

- In section: Timeout error,
    - Changed 'At the end of the ID, ocr_2 is loaded with 36 (maximum possible response space)' to 'At the end of the ID, OC2 is loaded with 36 (maximum possible response space) + LINFlexD_LINTCSR[CNT]'
    - Changed 'ocr_2' to 'OC2'

- In section: Memory map and register description,
    - Removed sentence "The reset value is the 2Dh = 45 ... the LIN specification." from the description of 'HTO' bit of register 'LINTOCR'
- In section: Interrupts, added note: "Rx interrupt due to ... LSIE is enabled." in the figure: 'Interrupt diagram'

- In section: Memory map and register description,
    - In register LINSR,
        - Changed the access of bit field 'LINS' to ROWO (was RO)
        - Added note 'The value of this bit field doesn’t change ... LIN state event. to the bit field description. '
    - In register, UARTSR, updated the bit field description of bit DRFRFE.
    - In register, UARTSR, updated the bit field description of bit DTFTFF.
    - In register LINTOCR, clarified the reset value note of bit field HTO.

- Updated section: Header error

- In section: Memory map and register description,
    - In register UARTCR, enhanced the bitfield description of bits WL0, WL1, PC0, and PC1.

- Updated section: Baud rate generation

- In section: Memory map and register description,
    - In register 'LINESR' updated the bitfield description of 'OCF'

- Added section: LINFlexD Clock Tolerance

• In section Memory map and register description:, updated note 'In Master Mode, read access to IFMI register and ...'
• In section Timeout error, heading Response timeout mechanism: • Changed 'Nominal_time_out-Correction_factor' to 'Nominal_time_out + LINFlexD_LINTCSR[CNT]'   • Added note 'The value of nominal_time_out ... '
• In section Memory map and register description: • Added note: "Any access to A0 location ... transfer error."
• In section: Timeout, figure: Timeout, changed OC1 to OC2 • In section: Timeout error heading: Header timeout mechanism, updated bullet: "If the counter starts after 11 ... " • Updated note: "The LINFlexD module ..." present in UARTCR register.
• Updated section:Timer • In section: Memory map and register description   • Updated the access of register UARTCTO to R in table: LINFlexD memory map   • Removed phrase 'free running' from bit LINTCSR[CNT].

B.53 ENET module changes

• Added Input Capture and Output Compare.
• Corrected field width of ENET_MRBR[R_BUF_SIZE].
• Corrected access type of ENET_ATCR[Reserved] to WOO and added note to its description that it must be written with one.
• In MDIO clause 22 frame format changed "IEEE803.2" to "IEEE 802.3". • Made improvements to Block diagram figure.
• Memory map/register definition : Added statement regarding allowed access width to registers and made editorial changes. • Transmit Inter-Packet Gap (ENET_TIPG) : Corrected description of IPG field. • Inter-packet gap (IPG) : Corrected text regarding IPG values. • Removed sentences regarding register reset/initialization in Physical Address Lower Register (ENET_PALR) and Physical Address Upper Register (ENET_PAUR). • Corrected references to specific bit numbers in ENET_RDSR, ENET_TDSR, and ENET_MRBR descriptions. • Improved descriptions of Statistic Registers fields. • Removed Receive Parser block from Figure 53-5.
• Updated COUNT field descriptions in Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK), Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL), and Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK). • Changed occurrences of DMA to uDMA, where appropriate, for clarity. • Added note to ENET_IEEE_T_FRAME_OK[COUNT] description. • Added note to ENET_IEEE_R_OCTETS_OK[COUNT] description. • Added receive and transmit FIFO depths to FIFO thresholds.
• Added description of uDMA to Overview. Replaced "DMA" with "uDMA" in Transmit FIFO Watermark Register (ENET_TFWR) and Soft reset. • Added sections for RMON_T_DROP, IEEE_T_DROP, IEEE_T_SQE, and RMON_R_RESVD_0 registers. • Added new IEEE 802.3 Clause 45 content: • Added this feature to Ethernet MAC features.   • Modified ENET_MMFR field descriptions.   • Changed content and title of MDIO clause 22 frame format.   • Added MDIO clause 45 frame format. • In ENET_TCSRn[TMODE} description, changed "1100 Reserved" to "110X Reserved". • External signal description : Corrected description of ENET_TCSR*n*[TIE] regarding interrupts. Corrected description of ENET_TCSR*n*[TIE] regarding DMA requests.
• Maximum Receive Buffer Size Register (ENET_MRBR) : Edited register and field descriptions for clarity regarding max. receive buffer size and how value is formed.

- Collision detection and handling — half-duplex operation only : Added Warning regarding use of Ethernet PHYs that support the "heartbeat" feature.

- Changed both MAC frame format overview figures in Ethernet MAC frame formats.
- Changed field description of ENET_ATCR[CAPTURE].
- Edited introductory text in Memory map/register definition.
- Edited description for Overrun in Enhanced receive buffer descriptor.
- Edited Overview and removed "10/100" in last paragraph.
- Editorial changes.

- Timer Period Register (ENET_ATPER) : Added note to PERIOD description about value.

- MII Speed Control Register (ENET_MSCR) :
  - Changed "internal module clock" to "internal module clock (i.e., IPS bus clock)".
  - Edited MII clock freqency calculation example.
  - In header of "Programming Examples for MSCR" table, changed "Internal MAC clock frequency" to "Internal module clock frequency".
- Receive FIFO overflow : In MII Receive portion of diagram, changed "MII_RXEN" to "MII_RXDV".
- MDIO clock generation : Changed "internal bus clock" to "internal bus clock (i.e., IPS bus clock)".

B.54 MC_RGM changes

- No substantial content changes

B.55 BAM changes

- No substantial content changes

B.56 SSCM module changes

- Updated Features list

- In CERS bit description, added "If a non-permanent error is detected, this bit can be cleared by writing a 1."

- In Table 56-8 updated 0:31 | 0100_000Ah to 0:31 | 0100_000Ch

- In Boot mode functionality in SBL bullet removed text "(support for the FlexRay interface is planned on future devices)".
- In Hardware configuration removed table column "standby RAM boot flag".
- In Securing the microcontroller added "DCF" to the table titles for Table 56-8, Table 56-9, and Table 56-10.

- In Boot and alternate boot sectors updated second bullet.

- In CER removed "(with PARITY enabled)" from the bit description.

- In CERS added cross-reference to CERS-DCF Mechanism.

- In SSCM System Status (SSCM_STATUS) register's CER field description, changed first sentence and added sentence:
  - was: "This field indicates that the SSCM has detected a configuration error during reset while loading DCF clients."
  - now: "This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error."

- Added section: CERS-DCF Mechanism

- In Memory map and register definition, added additional text ("that is, you should not..") to note regarding register accesses.
- Editorial changes.

- Revised CERS-DCF Mechanism.
- In CER, added cross-reference to CERS-DCF Mechanism.

- Note updated in Memory map and register definition from "SSCM does not support transfer errors in memory holes" to "SSCM does not generate transfer errors in all memory holes".

- Previous errata e8014 integrated into reference manual, Added a note "When external reset......... reset" in Unsecuring the microcontroller section.

- Removed sentence "To modify.....password" in Securing the microcontroller section.

B.57 PMC changes

- In Default Mode removed paragraph: "This startup self test function also provides gating of the reset phase3 exit in MC_RGM. This means the MC_RGM will not be able to exit from phase 3 until the self test is successfully completed."

- In Reset Event Selection 0 (PMC_RES_0) description, added second sentence to Note: "The user should configure a functional reset for an LVD/HVD only for debugging purposes and never for any actual application."

- For Event Status Register 0 (PMC_ESR_0)
  - Changed access of register from read/write to "write 1 to clear"
  - Changed access of every implemented field from read/write to "write 1 to clear"
  - Replaced VD0_C, VD15_C, VD12_C, VD5_C, VD2_C, and VD1_C with Reserved fields
- In Temperature Event Status register (PMC_ESR_TD) for TEMP1_3, TEMP1_2, TEMP1_0, TEMP0_3, TEMP0_2, and TEMP0_0 fields:
  - Changed access of these fields from read/write to "write 1 to clear"
  - In description of these fields, changed second sentence:
    - from: "It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a one is written to this bit location."
    - to: "Its value becomes 1 when the temperature exceeds its corresponding threshold, and that value clears to 0 when 1 is written to this location."
  - In description of these fields, removed sentences about field dependencies (following sample is from TEMP1_3 description): "If the IER[TS1_3IE] bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_3] is also asserted, a system reset is generated, which clears IER[TS1_3IE] and negate the interrupt request. If REE_TD[TEMP1_3] is asserted and a destructive reset is selected (via RES_TD[TEMP1_3] being 0), then a destructive reset is generated. If RES_TD[TEMP1_3] is set, then a functional reset is generated."
- In Temperature Event Status register (PMC_ESR_TD) for TEMP1_3_OP, TEMP1_2_OP, TEMP1_0_OP, TEMP0_3_OP, TEMP0_2_OP, and TEMP0_0_OP fields:
  - In description of these fields, added sentences about field dependencies that were removed from corresponding TEMPn_m description (following sample is from TEMP1_3_OP description): "If IER[TS1_3IE] is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_3] is also asserted, a system reset is generated, which clears IER[TS1_3IE] and negates the interrupt request. If REE_TD[TEMP1_3] is asserted and a destructive reset is selected because RES_TD[TEMP1_3] is 0, then a destructive reset is generated; if RES_TD[TEMP1_3] is 1, then a functional reset is generated."

- In Temperature Reset Event Enable register (PMC_REE_TD) description, added Note and changed second paragraph:
  - from: "These bits are loaded from flash memory during the boot sequence. If the values loaded from flash memory are 1, enabling the reset event, these bits cannot be cleared to 0 to disable the event."
  - to: "The register's reset value applies until the portion of the boot sequence that loads a value for the register from a DCF record in the UTEST area of flash memory. The DCF record for this register is PMC_LVD_MISC. The user can change the loaded value by programming a new DCF record for PMC_LVD_MISC in the UTEST

area. If the loaded value programs any flag bit to be enabled (set to 1), the bit cannot be disabled (cleared to 0) by software through IPS programming."

- In Temperature Reset Event Selection register (PMC_RES_TD) description, added new second paragraph: "The register's reset value applies until the portion of the boot sequence that loads a value for the register from a DCF record in the UTEST area of flash memory. The user can change the loaded value by programming a new DCF record in the UTEST area."

- In Reset Event Enable 0 (PMC_REE_0) description:
  - Removed from first paragraph: "These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared)."
  - Added new second paragraph: "The register's reset value applies until the portion of the boot sequence that loads a value for the register from a DCF record in the UTEST area of flash memory. The user can change the loaded value by programming a new DCF record in the UTEST area. If the loaded value programs any flag bit to be enabled (set to 1), the bit cannot be disabled (cleared to 0) by software through IPS programming."
- For following Reset Event Enable 0 (PMC_REE_0) fields, clarified voltage-detect event to which each field applies:
  - VD6RE_C
  - VD4RE_C
  - VD3RE_H
  - VD3RE_C

- For Event Status Register 0 (PMC_ESR_0) field VD4_C
  - Changed first sentence of field description from "This read-only bit is the low-voltage status flag associated with the voltage level detect for the low voltage cold point supply" to "This read-only bit is the high-voltage status flag associated with the voltage level detect for the high-voltage cold point supply."
  - In description of field value 1, changed "HVD occurrence detected on the low voltage supply" to "HVD occurrence detected on the high-voltage supply."

B.58 MC_PCU changes

- No substantial content changes

B.59 MC_ME changes

- Previous errata e6574 integrated into reference manual:
  - In HALT0 Mode, added a new paragraph "If a HALT0 mode transition request is generated at the same time that an interrupt request occurs ...."
  - In STOP0 Mode, added a new paragraph "If a STOP0 mode transition request is generated at the same time that an interrupt request occurs ......"

- In HALT0 Mode, added a new bullet "XBAR masters cannot operate in system HALT0 mode as they are clock gated. Also, NPC should be enabled if mode transition is initiated while debugger is connected."
- In STOP0 Mode, added a new bullet "XBAR masters cannot operate in system STOP0 mode as they are clock gated. Also, NPC should be enabled if mode transition is initiated while debugger is connected."

- Added a new section: Mode transition and progressive clock switching
- In System Clock Switching, updated paragraph "If the value of the PWRLVL field of the ME ..."

B.60 Core Debug Support changes

- No substantial content changes

B.61 JTAGC module changes

- No substantial content changes

B.62 Core Nexus Module changes

- Nexus Development Control Register 4 (DC4) :
  - Corrected broken link.
  - Removed footnote from "DC4 field descriptions" table.

- Replaced nex_evto_b with evto_b throughout chapter.

- Added missing TSTAMP sub-row to each Message Name row in Table 62-2 table.
- Data Acquisition Trace event : in figure, corrected "TCODE(011011)" to "TCODE(000111)".
- Added Timestamp field (TSTAMP).
- Data Trace Messaging (DTM) : Changed "in one of three ways." to "in one of the following ways."

B.63 NXMC module changes

- Previous errata e7974 integrated into the Reference Manual: Added Note in DTM overflow error messages, "The internal queue FIFO is very small. This causes FIFO overflows whenever Trace is enabled on any of the AHB clients and the AHB master module generates >4 burst accesses. If the customer is experiencing a lot of overflow error messages the the amount of data snooped by the NXMC should be reduced."

- Editorial change in DTM overflow error messages section

B.64 NPC module changes

- Updated PCR[FPM] description from "1 Reserved" to "1 Aurora mode enabled" .

- In Port Configuration Register (PCR), added "Nexus" to PCR[FPM] description.

- Updates throughout: removed various instances of "auxiliary", changed "auxiliary" to "output", changed "Auxiliary" to "Nexus" and removed text "As an alternative to data transmission on the Nexus auxiliary port".
- In Port Configuration Register (PCR) :
  - Added text "Set to 000 when Nexus Aurora mode is enabled" to PCR[MCKO_DIV] description.
  - Updated PCR[FPM] description.

- In Port Configuration Register (PCR), added text "While enabling the low power mode handshake (setting LP_DBG), the debugger should simultaneously set the LPn_SYN bit. At the end of low-power mode handshake, the LP_DBG and LPn_SYN should be set again if further handshakes between the device and debugger are required." to second paragraph.

B.65 NAL changes

- No substantial content changes

B.66 Nexus Aurora Phy changes

- No substantial content changes

B.67 CRC module changes

- Title corrected for section "Register programming".
- Added information about register access level at the end of each register description.
- Editorial changes.
- In CRC_CFG*n*[SWAP], updated bit description text from "The swap operation is a bit-by-bit rotation of the content" to "The swap operation is a bit-by-bit swapping of the content".
- In CRC_CFG*n*[INV], removed the bit-description text "in case of CRC-CCITT polynomial, the inversion operation is applied on the 16 least-significant bits.".
- Updated title of Signal description section to External signal description.
- In Block diagram, updated reference to Main features section to see the number of contexts of CRC.

B.68 MEMU changes

- In Memory map and register definition, added register access level information to all register descriptions.

- Previous errata e8145 integrated into this reference manual: In System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR), in the ERR_ADD field description, deleted "This register must be READ ONLY when its corresponding valid bit is not asserted."
- Previous errata e8145 integrated into this reference manual: In Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR), in the ERR_ADD field description, deleted "This register must be READ ONLY when its corresponding valid bit is not asserted."
- Previous errata e8145 integrated into this reference manual: In Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR), in the ERR_ADD field description, deleted "This register must be READ ONLY when its corresponding valid bit is not asserted."

- Editorial changes.
- In Introduction and Features :
  - Standardized the names of the error-reporting categories for consistency:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Changed "see the Device Configuration chapter that describes how modules are configured and connected" to "see the chip-specific MEMU information."

• In Design overview, changed "see the Device Configuration chapter that describes how modules are configured and connected" to "see the chip-specific MEMU information." • In Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*), in the BAD_BIT field description, changed "bit position in RAM" to "bit position in flash memory".
• Revised Figure 68-2 and Figure 68-3. • In System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR), added "This field must be read-only when its corresponding valid bit (SYS_RAM_UNCERR_STS[VLD]) = 0." to the ERR_ADD field description. • In Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR), added "This field must be read-only when its corresponding valid bit (PERIPH_RAM_UNCERR_STS[VLD]) = 0." to the ERR_ADD field description. • In Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR), added "This field must be read-only when its corresponding valid bit (FLASH_UNCERR_STS[VLD]) = 0." to the ERR_ADD field description.
• In Design overview, deleted "The MEMU block has three instantiations of the basic reporting block along with a common CPU interface for the user software to access the storage block.". • In Overview, deleted the paragraph beginning with "The Control, Error Flag, and Debug".

B.69 FCCU module changes

• EOUT interface   • Reorganized the sentence about resets' effect on failure indication pins as two sentences and a list   • In the list, added "FOSU time-out reset" as a reset type that influences the state of failure indication pins
• Changed reset value of FCCU_CTRL register from 0x0000_0000 to 0x0000_00C0. (OPS field changed from 00b to 11b.)
• In NCF Fake Register (FCCU_NCFF), changed register description.
• INTRODUCTION SECTION - Removed phrase "(after the boot)"
• INTRODUCTION SECTION - Change "Internal reactions (independently configurable for each NCF):" to "Configurable internal reactions for each NCF:" • INTRODUCTION SECTION - Change "External reaction (failure is reported to the outside world via one or more output pins)" to "External reactions via configurable output pins" • INTRODUCTION SECTION - Removed phrase "without CPU intervention". • INTRODUCTION SECTION - Removed phrase "for application/test/debugging purposes" • INTRODUCTION SECTION - Changed "FCCU is designed for assuming that the system clock is faster than the IRC clock" to "The FCCU is designed to function when the system clock is faster than the IRC clock." • FSM description - Change "After reset lifts, the state shall be transiently locked (permanent lock → is unlocked and transient lock → is locked)." to "After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked." • In the FCCU_CTRLK register description, change note "There should not be any other operation in between the above steps." to "The FCCU_CTRLK and FCCU_CTRL registers must be written with consecutive instructions. Both registers must be written as 32-bit values. Do not use read-modify-write instructions, such as bit field instructions, to modify these registers." • In section FCCU_DELTA_T, FCCU_IRQ_ALARM_ENn, FCCU_NMI_ENn, FCCU_EOUT_SIG_ENn descriptions: Changed, "This register can be written only when the FCCU is in CONFIG state." to Note. • In FCCU_CFG_TO description changed "oscillator" to "IRCOSC". Changed part of description to Note.
• For FCCU_NCF_CFGn, FCCU_NCFS_CFGn, FCCU_NCF_En, FCCU_NCF_TOEn and MCS registers: Changed the reset value of the registers from 0x0000_0000 to X. Added the following footnote to each register: The reset value is chip-specific. See the chip-specific FCCU information. Added the following note to register descriptions: The reset value is chip-specific. See the chip-specific FCCU information. For all configuration registers, made the following text into a note: These registers can be written only when the FCCU is in CONFIG state. • INTRODUCTION -- Changed bullet item "Configurable and graded fault control" to "Configurable fault control" • INTRODUCTION -- In bullet item "Permanently locked until next reset ...., Changed "FCCU_TRANS_LOCK" to "FCCU_PERMNT_LOCK"

• INTRODUCTION -- Changed "failure indication" to "error out" • MAIN features -- Changed "Failure indication" to "Error indication" • FSM description -- Changed reference to state diagram. • FSM description -- Changed "Following one of the following events" to "Following state transitions occur on one of the following event" • Corrected notation for binary numbers. • FCCU state diagram -- Changed block text on CONFIG state from "AND NOT (configuration locked)" to "AND (configuration unlocked)" • FCCU state diagram -- Added block text to ALARM state -- "Actual text in diagram is "Any Fault Pending AND FCCU_IRQ_ALARM_ENn"" • EOUT interface -- Throughout section changed uppercase "ERROR" to "FAULT".
• At the end of EOUT interface section, added NOTE on state of EOUT pads.
• FCCU_CTRL register, FILTER_WIDTH field -- Changed 10 bit description from "filters glitches up to 100 µs for 20 MHz IRC clock" to "filters glitches up to 100 µs". Changed 11 bit description from "filters glitches up to 100 µs for 20 MHz irc clock" to "filters glitches up to 100 µs"
• Added the following note to the FCCU_NCFS_CFGn and FCCU_NMI_ENn register descriptions: Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time. • The following text was added to the EIN1 bit description of the FCCU_EINOUT register -- While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[1] pin. The following text was added to the EIN0 bit description of the FCCU_EINOUT register -- While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[0] pin.
• Acronyms and Abbreviations section: Added IRCOSC entry into the Acronyms and Abbreviations table. • FCCU_CTRL register, removed "configure the glitch filter present on location given by FILTER_POSITION parameter" • FCCU_CTRL[DEBUG] description, changed ipg_debug to debug. • Added the note "The reset value is chip-specific. See the Chip Configuration chapter for details." at the beginning of the following sections: NCF Configuration Register (FCCU_NCF_CFGn), NCFS Configuration Register (FCCU_NCFS_CFGn), NCF Enable Register (FCCU_NCF_En) and NCF Time-out Enable Register (FCCU_NCF_TOEn)
• Minor non-technical grammatical corrections. • FCCU_CFG_TO register description - replaced "See FCCU register set table." with "The description for a configuration register will contain a NOTE that it is writable only in the CONFIG state."
• FCCU_CFG register, FOP bit field description -- added the following note: The FCCU_CFG register value after a CFG timeout will be 0x0000_0008.
• In Dual-rail protocol, corrected the error-phase portion of Figure 69-11.
• FCCU_STAT register, STATUS field -- Changed "UNKNOWN" to "Reserved" in bit field descriptions. • Use Cases and Limitations section, Misconfigurations heading -- removed "Software reset is optional. NOTE: Software reset is not required for all devices." • FCCU_CFG register, FOP field description -- Changed "EOUTirq = / (FOP + 1) ...." to "EOUTirq = / (FOPE:FOP + 1) ...."
• In Definitions, after the list of applicable reset types, added this information: *For more information on each type of reset, see the MC_RGM and reset chapters.*
• In the FCCU_CFG register description, rewrote the field descriptions for FOPE and FOP.
• In the FCCU_CFG_TO register description, added this note to the register's function description: *When the configuration watchdog timer expires, FCCU changes the value of this register to its reset value.* • In the FCCU_EINOUT register description, added this note to the description of the EOUT1 and EOUT0 fields: *When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value.* • In the FCCU_STAT register description, changed the access type of the ESTAT field from *RW* to *RO*.
• In FSM description, added the paragraph that begins with the sentence *Multiple faults can occur at the same time.*
• In FSM description, added the paragraph that begins with the phrase *After moving to FAULT state*. • In OPR, added statement about OP15.
• In the FCCU_CFG register description, in the description for the FCCU_SET_CLEAR field, changed *by writing into these fields with 00 or 10 or by going to CONFIG state* to *by writing into these fields with 00 or 10*.

• In the FCCU_CFG register description, for the FCCU_SET_CLEAR field, changed the meaning of the value 01 from *Error Pin goes in Faulty state* to *Both error signals are set to low*. • In the FCCU_CFG register description, for the FCCU_SET_CLEAR field, changed the meaning of the value 11 from *write: Error Pin goes OK state (write not possible when FCCU already in the T min timer phase or if FSM enters in FAULT state by capture of a fault)* to *Both error signals are set to high (write not possible when FCCU is already in the T_min timer phase or if FCCU enters Fault state by capture of a fault)*.
• In FSM description, added two paragraphs that begin with the phrase *All pending faults that occur during CONFIG state*.
• In Main features, changed *Fault detection collection* to *Fault detection and collection*. • In Block diagram, changed the stem sentence for the table to *This table describes the FCCU submodules.*
• In FCCU Output Supervision Unit, changed the first sentence of the third paragraph to *There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction for ALL FAULTS*
• In Definitions, changed the descriptions of the reset types from a list to a table, clarified the description of the destructive reset, and corrected the descriptions of the long and short functional resets.
• In the FCCU_CFG register description, added this note: *If you specify a new value for any of the fields in this register that affect the EOUT signals while the EOUT fault-indication timer is running (FCCU is indicating a fault on the EOUT signals), FCCU doesn't use the new settings you specified until after the EOUT fault-indication timer expires (FCCU stops indicating a fault on the EOUT signals).*
• Changed all references to the IRCOSC and RCOSC clocks so they are in terms of CLKSAFE, which represents either CLKSAFE0 or CLKSAFE1, both of which appear on the boundary of the module. • In EOUT interface, deleted this sentence: *The EOUT frequency is generated by dividing the CLKSAFE frequency by a fixed factor of* 2^{18}.
• Rewrote Signals (previously titled "Pinout"). • In the FCCU_DELTA_T register description, rewrote the field description for DELTA_T.
• In Introduction, changed *when the system clock is faster than the IRC clock* to *when CLKSYS is faster than the CLKSAFE clocks*. • In Acronyms and abbreviations, removed the row for IRCOSC. • In Acronyms and abbreviations, added a row for CF. • Rewrote NMI/WKPU interface. • Revised Figure 69-6 to use signal abbreviations that match those used in NMI/WKPU interface and now described in Signals. Also changed *All NMI signals are shown active-low* to *All NMI signals shown are active-low*.
• In Figure 69-14, added the following note: *The fault sources shown are examples only.*
• In the Noncritical Fault Timeout (FCCU_NCF_TO) register description, deleted the register reset value and replaced it with a footnote to indicate that it is chip-specific. Also corrected the register access type to *RW* and edited the register description for grammar and style. • In the Noncritical Fault Configuration (FCCU_NCF_CFG*n*) register description, deleted the register reset value and replaced it with a footnote to indicate that it is chip-specific. Deleted the following redundant note: *The reset value is chip-specific. See the chip-specific FCCU information.* Edited the register description for grammar and style.
• In the FCCU_MCS register description, changed the reset value of the register to 0 with a footnote to indicate that the MC_ME module changes the value of the register immediately after the chip leaves RESET mode. Rewrote the rest of the register description for clarity.
• For Reserved field of Configuration (FCCU_CFG)   • Changed the field access from read-only and always zero to read/write.   • In field description, added text: "Always write the reset value to this field."
• In the FCCU_CFG register description, added a footnote to the diagram to indicate that the register reset value is chip-specific. Deleted a similar note in the function description. Changed the access type of some reserved fields. • In the Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*) register description, changed the register access type to *R/W*. • In the FCCU_CFG_TO register description, deleted the register reset value and replaced it with a footnote to indicate that it is chip-specific.
• In the FCCU_STAT register description, in the description of the ESTAT field, deleted this paragraph: *SW can write this bit with a different value. The new value is valid for 1 clock cycle, thereafter it returns to indicate the actual value being driven from FCCU. For example in non-faulty states, this bit reads 0 and can be set for one cycle to 1. Similarly*

in faulty state, this bit reads 1 and SW can write it to 0 and after one cycle the bit reads 1 again. Edited the ESTAT description for style.

- In the FCCU_CTRL register description, rewrote the description of the DEBUG field.
- In Register descriptions, replaced the two paragraphs that begin with *The FCCU generates an access error in the following cases* with a paragraph that begins with *Follow these register-access guidelines*.

- In Use cases and limitations, changed all occurrences of *Tmin* to *T_min*.
- In the FCCU_DELTA_T register description, added this note to the end of the description of the DELTA_T field: *The specified values for DELTA_T may vary ±2% due to the post-trimming inaccuracy of the CLKSAFE clocks.*
- In Introduction, removed this sentence from the list because it duplicates information in the description of FCCU_DELTA_T[DELTA_T] and provides the wrong maximum value for DELTA_T: *T_min = 250 µs + delta_T, with delta_T parameter being configurable by SW up to 16.67 ms*.
- In STCU interface, changed *This feature is optionally programmable inside the STCU.* to *This feature is optionally programmable using MC_RGM.*

- In Introduction :
  - In the sublist for the list item *Configurable internal reactions for each NCF*, changed *No reset reaction* to *No reaction*.
  - Changed the list item *After power-on the error out pins have high impedance. FCCU goes to operational state only on software request.* to *After power on, the EOUT signals have high impedance. They show operational state only on software request.*

- In Definitions, in the list item *HW recoverable fault*:
  - Changed *until the fault cause is asserted* to *until the fault cause is deasserted*.
  - Changed *That is, if 0 on the fault signal indicates fault, then the status flags are valid until the fault line is '0'* to *That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line stays at 0*.
  - Changed *Typically the fault signal is latched in an external module at the FCCU* to *Typically the fault signal is latched external to the FCCU in the module where the fault occurred*.

- In FSM description :
  - Edited the content for style and usage.
  - In the list item *Configuration*, changed *The configuration state is used only to modify the default configuration of the FCCU* to *Used only to modify the configuration of FCCU from its default*.
  - In the stem sentence for the second list, changed *The transition from NORMAL/ALARM state goes along with the generation of* to *The transition from Normal/Alarm to Fault state goes along with the generation of*.

- In section Block diagram, Figure 69-1 :
  - Corrected bidirectional arrow between "FAULT" and block "FAULT intf" to a unidirectional arrow from "FAULT" to "FAULT intf".
  - Added "IRQ" to text label "MC_RGM, NMI"
- In section FSM description in the second paragraph of the first bullet, modified sentence "The permanent configuration lock can be disabled by a reset of the FCCU".
- In section FCCU Output Supervision Unit :
  - In the third paragraph, corrected "an enabled fault" to "a disabled fault".
  - In the fourth paragraphs, corrected "the FCCU does not check..." to "the FOSU does not check..."

- In EOUT interface :
  - Edited the section for style, usage, grammar, clarity, and structure, which included reordering some information and moving some information to the appropriate register descriptions.
  - Changed *Destructive reset* to *Destructive reset (including when an FOSU timeout occurs due to FCCU failing to react)*.
  - Deleted the list item *FOSU timeout reset*.
  - Changed *The FCCU provides up to two bidirectional signals (EOUT interface)* to *The FCCU provides up to two bidirectional signals (EOUT[1:0])*.
  - Changed *These signals need to be mapped to appropriate pins for external visibility* to *For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information*.
  - In the list that begins with *These resets influence the state of the failure indication pins*, changed *Pin reset (RESET_b)* to *Chip pin reset (RESET_B)*.

- In the FCCU_CFG register description, edited the following field names and descriptions for clarity:
  - FOPE
  - SM

• PS • FOM • FOP
• Rewrote the FCCU_EOUT_SIG_EN register description for clarity. • Added the following note to the FCCU_EOUT_SIG_EN[EOUTENx] field description: *To ensure that the FOSU module is aware that fault-output signaling is enabled as a reaction for the associated noncritical fault channel (so that FOSU doesn't mistakenly generate a destructive chip reset), you must set this field to enabled when FCCU is configured for any mode other than Bistable fault-output mode.*
• In FSM description, in the list item *Configuration*, added the following: *The locking feature only restricts the FSM movement into Configuration state. Once the user enters Configuration state and then tries to lock the configuration by writing either to TRANS_LOCK or PERMT_LOCK, the locking of configuration will be effective after FCCU moves to Normal state, it will not be effective in the current Configuration state.*

B.70 STCU2 changes

• In Startup self-test mode selection, added Note: "During startup self-tests, once the STCU2 registers are configured they cannot be overridden." • In "Shutdown self-test initiation procedure, added Note: "During shutdown self-tests, once the STCU2 registers are configured they cannot be overridden via IPS until the self-testing is complete."
• Added modules for Partition C0 in Table 70-6. • Added a note "Make sure that the flash is in IDLE state before starting LBIST (Self Test) operation." in Use cases and limitations.
• In Table 70-6, added RCCU module in Partition C1 and ENET module in Partition P0.
• Updated STCU online configuration in Table 70-4.

B.71 REG_PROT module changes

• In Memory map and register definition, made editorial changes.
• In Overview, added AIPS_LITE as an option to use the term PBRIDGE (PBRIDGE/ AIPS_LITE). • In General, modified the sentence "For all addresses that are protected …" to "For all addresses (peripheral registers) that are protected ….". • Removed topic "Initialization/application information".

B.72 WKPU module changes

• Editorial changes.
• In NMI Configuration Register (WKPU_NCR), changed the field name in NCR[NDSS0] note from HID0[MCP_EN] to HID0[EMCP].
• In WKPU memory map and register definition, modified the note "Reserved registers will read...". • In NMI management, added a sentence "Please see chip specific section for details on NMI implementation".
• Added register access level information.
• In Non-maskable interrupts, changed "The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event" to "The WKPU supports the capturing of a

second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event, though it creates an overrun condition."

- Changed NMI Status Flag Register (WKPU_NSR) access type from read/write to W1C.
- In External signal description, deleted a paragraph.

B.73 Protected Registers changes

- No substantial content changes

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Document Number MPC5744PRM
Revision 6, 06/2016