TMS570LS0232 16/32-Bit RISC Flash Microcontroller

Technical Reference Manual

Literature Number: SPNU603B
March 2018

Contents

List of Figures

List of Tables

Read This First

About This Manual

This Technical Reference Manual (TRM) details the integration, the environment, the functional description, and the programming models for each peripheral and subsystem in the device.

The TRM should not be considered a substitute for the data manual, rather a companion guide that should be used alongside the device-specific data manual to understand the details to program the device. The primary purpose of the TRM is to abstract the programming details of the device from the data manual. This allows the data manual to outline the high-level features of the device without unnecessary information about register descriptions or programming models.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers may be shown with the suffix h or the prefix 0x. For example, the following number is 40 hexadecimal (decimal 64): 40h or 0x40.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties with default reset value below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure can have one of multiple meanings:
    - Not implemented on the device
    - Reserved for future device expansion
    - Reserved for TI testing
    - Reserved configurations of the device that are not supported
  - Writing nondefault values to the Reserved bits could cause unexpected behavior and should be avoided.

Glossary

TI Glossary —This glossary lists and explains terms, acronyms, and definitions.

Related Documentation From Texas Instruments

For product information, visit the Texas Instruments website at http://www.ti.com.

SPNS242— *TMS570LS0232 16- and 32-Bit RISC Flash Microcontroller Data Manual.*

SPNU552— *Safety Manual for TMS570LS04x/03x Hercules™ ARM® Safety Critical Microcontrollers User's Guide.* A safety manual for the Texas Instruments Hercules safety critical microcontroller product family. The product family utilizes a common safety architecture that is implemented in multiple application focused products.

SPNU569— *TMS570LS04x Hercules™ Development Kit (HDK) User's Guide.* Describes the board level operations of the TMS570LS04 Hercules Development Kit (HDK). The HDK is based on the Texas Instruments TMS570LS0432 Microcontroller. The TMS570LS04 HDK is a table top card that allows engineers and software developers to evaluate certain characteristics of the TMS570LS0432 microcontroller to determine if the microcontroller meets the designer application requirements. Evaluators can create software to execute on board or expand the system in a variety of ways.

Community Resources

The following links connect to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

TI E2E™ Online Community— ***TI's Engineer-to-Engineer (E2E) Community.*** Created to foster collaboration among engineers. At e2e.ti.com, you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

TI Embedded Processors Wiki— ***Texas Instruments Embedded Processors Wiki.*** Established to help developers get started with Embedded Processors from Texas Instruments and to foster innovation and growth of general knowledge about the hardware and software surrounding these devices.

Trademarks

Introduction

Topic Page

1.1 Designed for Safety Applications

The TMS570LS0232 device architecture has been designed from the ground up to simplify development of functionally safe systems. The basic architectural concept is known as a safe island approach. Power, clock, reset, and basic processing function are protected to a high level of diagnostic coverage in hardware. Some of the key features of the safe island region are:

- A dual core lockstep processing solution built around ARM® Cortex®-R4 CPU that detects failures at the core boundary on a cycle by cycle basis. Special measures in processor layout, clock distribution, power distribution, reset distribution, and temporal diversity are all implemented to mitigate common cause failures of the logical CPU and its checker. For complete details on the ARM® Cortex®-R4 CPU, refer to the ARM® Cortex®-R4 Technical Reference Manual.
- Hardware BIST controllers which provide an extremely high level of diagnostic coverage for the lockstep CPUs and SRAMs in the system, while executing faster and consuming less memory than equivalent software-based self-test solutions
- ECC on the SRAM and flash memories are tightly coupled to the R4. The ECC controllers are located inside the CPU. This approach has two key advantages:
  - The interconnect between the CPU and the memory is also covered by the diagnostic
  - The ECC logic itself is checked on a cycle by cycle basis
- Onboard voltage and reset monitoring logic
- Onboard oscillator and PLL failure detection logic including a backup RC oscillator that can be utilized upon failure

The TMS570LS0232 device architecture also includes many features to simplify diagnostics of remaining logic such as:

- Continuous parity diagnostics on all peripheral memories
- Analog and digital loopback to test for shorts on I/O
- HW self-test and diagnostics on the ADC module to check integrity of both analog inputs and the ADC core conversion function
- A hardware engine for the background calculation of CRC signatures during data transfers
- A centralized error reporting function including a status output pin to enable external monitoring of the device status

1.2 Family Description

The TMS570LS0232 integrates the ARM® Cortex®-R4 CPU that offers an efficient 1.66 DMIPS/MHz and has configurations that can run up to 80MHz providing up to 132 DMIPS. The device supports the big-endian [BE32] format.

The TMS570LS0232 has up to 128KB integrated Flash and up to 32KB data RAM with single-bit error correction and double bit error detection. The flash memory on this device is a nonvolatile, electrically erasable and programmable memory implemented with a 64-bit-wide data bus interface. The flash operates on a 3.3V supply input (same level as I/O supply) for all read, program and erase operations. When in pipeline mode, the flash operates with a system clock frequency of up to 80MHz. The SRAM supports single-cycle read/write accesses in byte, halfword, and word modes.

The TMS570LS0232 device features peripherals for real-time control-based applications, including two Next Generation High End Timer (N2HET) timing coprocessors with up to 19 total IO terminals and a 12-bit A to D converter supporting up to 16 inputs.

The N2HET is an advanced intelligent timer that provides sophisticated timing functions for real-time applications. The timer is software-controlled, using a reduced instruction set, with a specialized timer micromachine and an attached I/O port. The N2HET can be used for pulse width modulated outputs, capture or compare inputs, or general-purpose I/O. It is especially well suited for applications requiring multiple sensor information and drive actuators with complex and accurate time pulses. A High End Timer Transfer Unit (HET-TU) can perform DMA type transactions to transfer N2HET data to or from main memory. A Memory Protection Unit (MPU) is built into the HET-TU.

The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine as used in high-performance motion and position-control systems.

The device has a 12-bit-resolution MibADC with 16 total channels and 64 words of parity protected buffer RAM. The MibADC channels can be converted individually or can be grouped by software for sequential conversion sequences. There are three separate groupings. Each sequence can be converted once when triggered or configured for continuous conversion mode.

The device has multiple communication interfaces: one MibSPI, two SPIs, one LIN/SCI, and two DCANs. The SPI provides a convenient method of serial interaction for high-speed communications between similar shift-register type devices. The LIN/SCI supports the Local Interconnect standard 2.0 and can be used as a UART in full-duplex mode using the standard Non-Return-to-Zero (NRZ) format. The DCAN supports the CAN 2.0B protocol standard and uses a serial, multimaster communication protocol that efficiently supports distributed real-time control with robust communication rates of up to 1 megabit per second (Mbps). The DCAN is ideal for applications operating in noisy and harsh environments (for example, automotive and industrial fields) that require reliable serial communication or multiplexed wiring.

The frequency-modulated phase-locked loop (FMPLL) clock module is used to multiply the external frequency reference to a higher frequency for internal use. The FMPLL provides one of the seven possible clock source inputs to the global clock module (GCM). The GCM module manages the mapping between the available clock sources and the device clock domains.

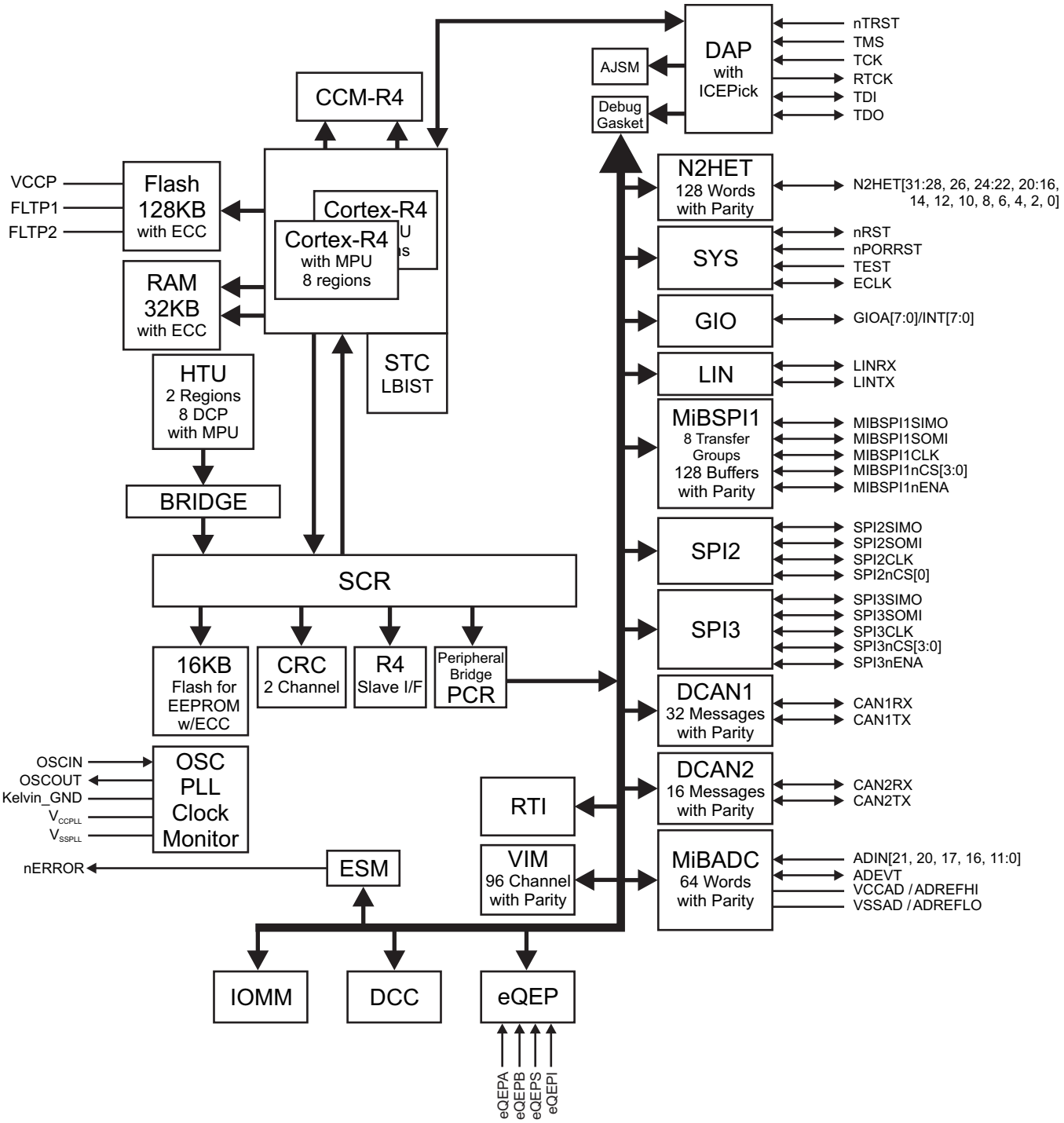
The device also has an external clock prescaler (ECP) module that, when enabled, outputs a continuous external clock on the ECLK pin. The ECLK frequency is a user-programmable ratio of the peripheral interface clock (VCLK) frequency or oscillator (OSCIN). This low frequency output can be monitored externally as an indicator of the device operating frequency.

The Error Signaling Module (ESM) monitors all device errors and determines whether an interrupt or external Error pin/ball is triggered when a fault is detected. The nERROR can be monitored externally as an indicator of a fault condition in the microcontroller.

The device supports all the built-in ARM® Cortex®-R4 CoreSight™ debug features.

With integrated safety features and a wide choice of communication and control peripherals, the TMS570LS0232 is an ideal solution for real time control applications with safety critical requirements.

Figure 1-1. Block Diagram



1.3 Endianism Considerations

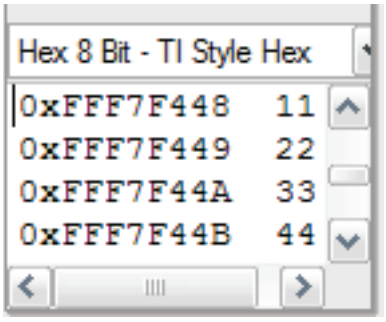
1.3.1 Big Endian (BE32)

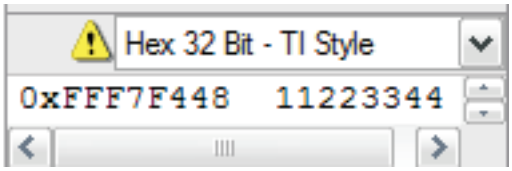
The TMS570 family is based on the ARM® Cortex®-R4 core. ARM has designed this core to be used in big-endian and little-endian systems. For the TI TMS570 family, the endianness has been configured to BE32. Big-endian systems store the most-significant byte of a multi-byte data field in the lowest memory address. Also, the address of the multi-byte data field is the lowest address. Below is an example of the physical addresses of individual bytes.

Figure 1-2. Example: SPIDELAY – FFF7 F448h

31 24	23 16
C2TDELAY[7:0]	T2CDELAY[7:0]
Byte 3 - FFF7 F448h	Byte 2 - FFF7 F449h
15 8	**7 0**
T2EDELAY[7:0]	C2EDELAY[7:0]
Byte 1 - FFF7 F44Ah	Byte 0 - FFF7 F44Bh

32-bit accesses to this register should use the lowest address, that is, 0xFFF7F448. Writing 0x11223344 to address 0xFFF7F448 shows the following when viewing the memory in 8-bit and 32-bit modes.





As such the headers provided as part of HALCoGen do take the endianness into account and provide header structures that are agnostic to endianness. This is achieved by using C directives for the compiler that make use of the compile options configured for the project by the user (__little_endian__ used in Code Composer Studio codegen tools). This directive may need to be adapted for other compilers.

```
#ifdef __little_endian__
    char  C2EDELAY      :  8U;  /**lt; 0xF448: CS to ENA        */
    char  T2EDELAY      :  8U;  /**lt; 0xF449: Transmit to ENA  */
    char  T2CDELAY      :  8U;  /**lt; 0xF44A: Transmit to CS   */
    char  C2TDELAY      :  8U;  /**lt; 0xF44B: CS to Transmit   */
#else
    char  C2TDELAY      :  8U;  /**lt; 0xF448: CS to Transmit   */
    char  T2CDELAY      :  8U;  /**lt; 0xF449: Transmit to CS   */
    char  T2EDELAY      :  8U;  /**lt; 0xF44A: Transmit to ENA  */
    char  C2EDELAY      :  8U;  /**lt; 0xF44B: CS to ENA        */
```

Architecture

This chapter consists of five sections. The first section describes specific aspects of the device architecture. The second section describes the clocking structure of the microcontrollers. The third section gives an overview of the device memory organization. The fourth section details exceptions on the device, and the last section describes the system and peripheral control registers of the microcontrollers.

Topic **Page**

2.1 Introduction

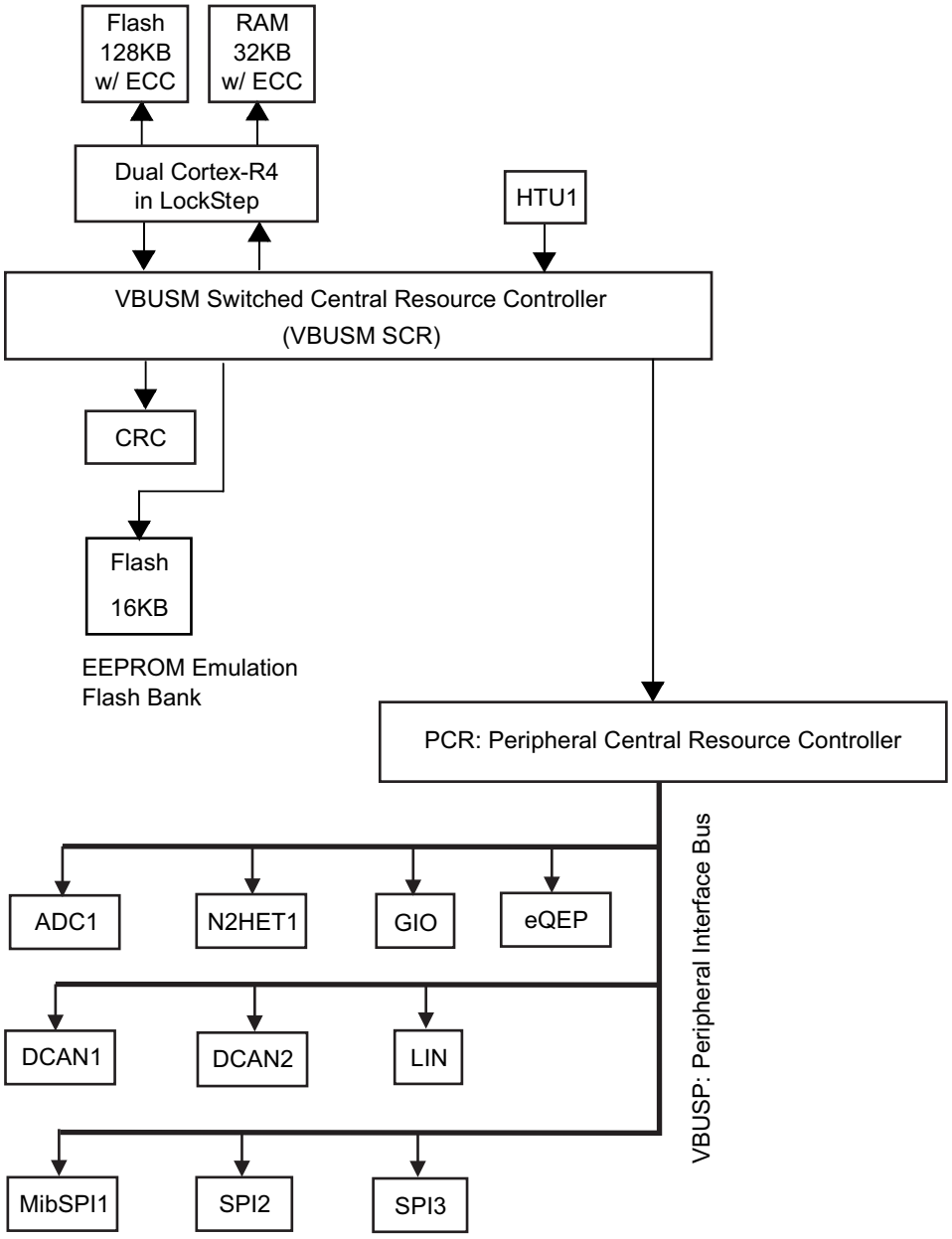
The TMS570LS0232 family of microcontrollers is based on the Texas Instruments TMS570 Architecture. This chapter describes specific aspects of the architecture as applicable to the TMS570LS0232 family of microcontrollers.

2.1.1 Architecture Block Diagram

The TMS570LS0232 microcontrollers are based on the TMS570 Platform architecture, which defines the interconnect between the bus masters and the bus slaves.

Figure 2-1 shows a high-level architectural block diagram for the superset microcontroller.

Figure 2-1. Architectural Block Diagram



2.1.2 Definitions of Terms

Table 2-1 provides a definition of terms used in the architectural block diagram.

Table 2-1. Definition of Terms

Acronym / Term	Full Form	Description
ADCx	Analog-to-Digital Converter	The ADC uses the Successive Approximation Register architecture. It features a selectable 10-bit or 12-bit resolution. The ADC module also includes a RAM to hold the conversion results. A digital logic wrapper manages accesses to the control and status registers. There is one ADC module on this device.
CRC	Cyclic Redundancy Checker	The CRC module provides two channels to perform background signature verification on any memory region. It also supports maximum-length Parallel Signature Analysis (PDS) based on a 64-bit primitive polynomial. The CRC module is a bus slave in this device.
DAP	Debug Access Port	The DAP allows a tool such as a debugger to read from or write to any region in the device memory-map. The DAP is a bus master in this device.
DCANx	Controller Area Network controller	The DCAN supports the CAN 2.0B protocol standard and uses a serial, multi-master communication protocol that efficiently supports distributed real-time control with robust communication rates of up to 1 megabit per second (Mbps). The DCAN is ideal for applications operating in noisy and harsh environments (for example, automotive and industrial fields) that require reliable serial communication or multiplexed wiring.
eFuse	Electronically Programmable Fuse controller	Electrically programmable fuses (eFuses) are used to configure the device after deassertion of PORRST. The eFuse values are read and loaded into internal registers as part of the power-on-reset sequence. The eFuse values are protected with Single-Bit Error Correction Double-Bit Error Detection (SECDED) codes. These fuses are programmed during the initial factory test of the device. The eFuse controller is designed so that the state of the eFuses cannot be changed once the device is packaged.
eQEP	Enhanced Quadrature Encoder Pulse Module	The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system.
ECC	Error Correction Code	This is a code that is used by the Single-Bit Error Correction Double-Bit Error Detection (SECDED) logic inside the two Cortex-R4 processors (CPUs). There are 8 bits of ECC for every 64 bits of data accessed from the CPU tightly-coupled memories (flash and RAM).
EEPROM Emulation Flash Bank	Emulated Electrically Erasable Programmable Read-Only Memory	This is a flash bank that is dedicated for use as an emulated EEPROM. This device supports 16KB of flash for emulated EEPROM.
GIO	General-purpose Input/Output	The GIO module allows up to 16 terminals to be used as general-purpose Input or Output. Each of these are also capable of generating an interrupt to the CPU.
HTUx	High-End Timer Transfer Unit	The HTU is a dedicated transfer unit for the New Enhanced High-End Timer module. The HTU has a native interface to the N2HET RAM, and is used to transfer data to / from the N2HET RAM from / to another region in the device memory-map. There is one HTU and one N2HET module on the device. The HTUx are bus masters in this device.
LIN	Local Interconnect Network controller	The LIN module supports the Local Interconnect standard revision 2.1 and can be used as a UART in full-duplex mode using the standard Non-Return-to-Zero (NRZ) format.
Lockstep	–	This is the mode of operation of the dual ARM Cortex-R4 CPUs. The outputs of the two CPUs are compared on each CPU clock cycle. Any miscompare is flagged as an error of the highest severity level.
MibSPIx	Multi-Buffered Serial Peripheral Interface	The MibSPIx modules also support the standard SPI communication protocol. The transfers are all grouped into transfer chunks called “transfer groups”. These transfer groups are made up of one ore more buffers in the MibSPIx RAM. The RAM is used to hold the control information and data to be transmitted, as well as the status information and data that is received. There is one MibSPI module in this device.
N2HETx	New Enhanced High-End Timer	The N2HET is an advanced intelligent timer that provides sophisticated timing functions for real-time applications. The timer is software-controlled, using a reduced instruction set, with a specialized timer micromachine and an attached I/O port. The N2HET can be used for pulse width modulated outputs, capture or compare inputs, or general-purpose I/O.

Table 2-1. Definition of Terms (continued)

Acronym / Term	Full Form	Description
PCR	Peripheral Central Resource controller	The PCR manages the accesses to the peripheral registers and peripheral memories. It provides a global reset for all the peripherals. It also supports the capability to selectively enable or disable the clock for each peripheral individually. The PCR also manages the accesses to the system module registers required to configure the device's clocks, interrupts, and so on. The system module registers also include status flags for indicating exception conditions – resets, aborts, errors, interrupts.
SPIx	Serial Peripheral Interface	The SPIx modules provide a clocked serial communication interface for reliable communication between the device and other serial devices with the standard SPI interface. There are two SPI modules on this device.
VBUSM SCR	VBUSM Switched Central Resource controller	This is the main device SCR. It arbitrates between the accesses from multiple bus masters to the bus slaves using a round robin priority scheme.

2.1.3 Bus Master / Slave Access Privileges

This device implements some restrictions on the bus slave access privileges in order to improve the overall throughput of the interconnect shown in Figure 2-1.

Table 2-2. Bus Master / Slave Access Privileges

			Bus Slaves Being Accessed			
Masters	Master ID	Access Mode	EEPROM Bank, ECC Bits, OTP Regions	Non-CPU Accesses to CPU Flash and RAM	CRC Module	PCR Modules
CPU Read	0	User/Privilege	Allowed	Allowed	Allowed	Allowed
CPU Write	1	User/Privilege	Not allowed	Allowed	Allowed	Allowed
HTU	6	Privilege	Not allowed	Allowed	Allowed	Allowed

2.2 Memory Organization

2.2.1 Memory-Map Overview

The Cortex-R4 uses a 32-bit address bus, giving it access to a memory space of 4GB. This space is divided into several regions, each addressed by different memory selects. Figure 2-2 shows the superset memory-map of the microcontroller.

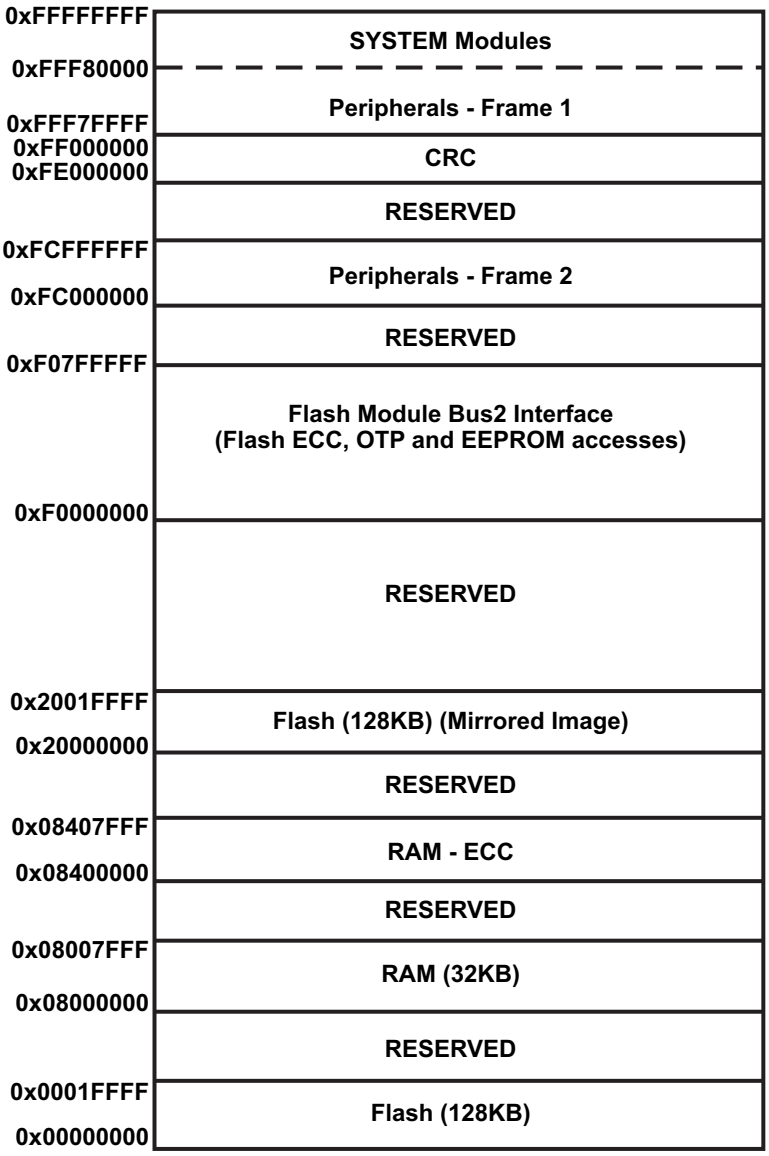
The main flash instruction memory is addressed starting at 0x00000000 by default. This is also the reset vector location – the ARM Cortex-R4 processor core starts execution from the reset vector address of 0x00000000 whenever the core gets reset.

The CPU data RAM is addressed starting at 0x08000000 by default.

The device also supports the swapping of the CPU instruction memory (flash) and data memory (RAM). This can be done by configuring the MEM SWAP field of the Bus Matrix Module Control Register 1 (BMMCR1).

After swapping, the data RAM is accessed starting from 0x00000000 and the RAM ECC locations are accessed starting from 0x00400000. The flash memory is now accessed starting from 0x08000000.

Figure 2-2. Memory-Map



2.2.2 Memory-Map Table

The control and status registers for each module are mapped within the CPU's 4GB memory space. Some modules also have associated memories, which are also mapped within this space.

Table 2-3 shows the starting and ending addresses of each module's register frame and any associated memory. The table also shows the response generated by the module or the interconnect whenever an access is made to an unimplemented location inside the register or memory frame.

Table 2-3. Module Registers / Memories Memory-Map

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
CPU Tightly-Coupled Memories						
TCM Flash	CS0	0x0000_0000	0x00FF_FFFF	16MB	128KB	Access above 0x3FFFF generates an Abort. (1)
TCM RAM + RAM ECC	CSRAM0	0x0800_0000	0x0BFF_3FFF	64MB	32KB	
Mirrored Flash	Flash mirror frame	0x2000_0000	0x20FF_FFFF	16MB	128KB	
Flash Bus2 Interface: OTP, ECC, EEPROM Bank						
Customer OTP, TCM Flash Bank		0xF000_0000	0xF000_FFFF	64KB	2KB	Abort
Customer OTP, EEPROM Bank		0xF000_E000	0xF000_FFFF	8KB	1KB	
Customer OTP–ECC, TCM Flash Bank		0xF004_0000	0xF004_03FF	1KB	512B	
Customer OTP–ECC, EEPROM Bank		0xF004_1C00	0xF004_1FFF	1KB	128B	
TI OTP, TCM Flash Bank		0xF008_0000	0xF008_FFFF	64KB	2KB	
TI OTP, EEPROM Bank		0xF008_E000	0xF008_FFFF	8KB	1KB	
TI OTP–ECC, TCM Flash Bank		0xF00C_0000	0xF00C_03FF	8KB	512B	
TI OTP–ECC, EEPROM Bank		0xF00C_1C00	0xF00C_1FFF	1KB	128B	
EEPROM Bank–ECC		0xF010_0000	0xF013_FFFF	256KB	2KB	
EEPROM Bank		0xF020_0000	0xF03F_FFFF	2MB	16KB	
Flash Data Space ECC		0xF040_0000	0xF04F_FFFF	1MB	48KB	
Cyclic Redundancy Checker (CRC) Module Register Frame						
CRC	CRC frame	0xFE00_0000	0xFEFF_FFFF	16MB	512B	Accesses above 0x200 generate an Abort.
Peripheral Memories						
MIBSPI1 RAM	PCS[7]	0xFF0E_0000	0xFF0F_FFFF	128KB	2KB	Abort for accesses above 2KB
DCAN2 RAM	PCS[14]	0xFF1C_0000	0xFF1D_FFFF	128KB	2KB	Wrap around for accesses to unimplemented address offsets lower than 0x7FF. Abort generated for accesses beyond offset 0x800.
DCAN1 RAM	PCS[15]	0xFF1E_0000	0xFF1F_FFFF	128KB	2KB	Wrap around for accesses to unimplemented address offsets lower than 0x7FF. Abort generated for accesses beyond offset 0x800.

(1) Address locations from 0x20000 to 0x3FFFF should be protected using the MPU to generate an access error upon accidental access; otherwise, there is no indication of accidental access to that memory region.

Table 2-3. Module Registers / Memories Memory-Map (continued)

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
MIBADC RAM	PCS[31]	0xFF3E_0000	0xFF3F_FFFF	128KB	8KB	Wrap around for accesses to unimplemented address offsets lower than 0x1FFF.
MIBADC Look-Up Table					384B	Look-up table for ADC wrapper. Starts at offset 0x2000 and ends at 0x217F. Wrap around for accesses between offsets 0x180 and 0x3FFF. Aborts generated for accesses beyond 0x4000.
N2HET RAM	PCS[35]	0xFF46_0000	0xFF47_FFFF	128KB	16KB	Wrap around for accesses to unimplemented address offsets lower than 0x3FFF. Abort generated for accesses beyond 0x3FFF.
HET TU RAM	PCS[39]	0xFF4E_0000	0xFF4F_FFFF	128KB	1KB	Abort
Debug Components						
CoreSight Debug ROM	CSCS0	0xFFA0_0000	0xFFA0_0FFF	4KB	4KB	Reads return zeros, writes have no effect
Cortex-R4 Debug	CSCS1	0xFFA0_1000	0xFFA0_1FFF	4KB	4KB	Reads return zeros, writes have no effect
Peripheral Control Registers						
HTU	PS[22]	0xFFF7_A400	0xFFF7_A4FF	256B	256B	Reads return zeros, writes have no effect
N2HET	PS[17]	0xFFF7_B800	0xFFF7_B8FF	256B	256B	Reads return zeros, writes have no effect
GIO	PS[16]	0xFFF7_BC00	0xFFF7_BCFF	256B	256B	Reads return zeros, writes have no effect
MIBADC	PS[15]	0xFFF7_C000	0xFFF7_C1FF	512B	512B	Reads return zeros, writes have no effect
DCAN1	PS[8]	0xFFF7_DC00	0xFFF7_DDFF	512B	512B	Reads return zeros, writes have no effect
DCAN2	PS[8]	0xFFF7_DE00	0xFFF7_DFFF	512B	512B	Reads return zeros, writes have no effect
LIN	PS[6]	0xFFF7_E400	0xFFF7_E4FF	256B	256B	Reads return zeros, writes have no effect
MibSPI1	PS[2]	0xFFF7_F400	0xFFF7_F5FF	512B	512B	Reads return zeros, writes have no effect
SPI2	PS[2]	0xFFF7_F600	0xFFF7_F7FF	512B	512B	Reads return zeros, writes have no effect
SPI3	PS[1]	0xFFF7_F800	0xFFF7_F9FF	512B	512B	Reads return zeros, writes have no effect
EQEP	PS[25]	0xFFF7_9900	0xFFF7_99FF	256B	256B	Reads return zeros, writes have no effect
EQEP (Mirrored)	PS2[25]	0xFCF7_9900	0xFCF7_99FF	256B	256B	Reads return zeros, writes have no effect

Table 2-3. Module Registers / Memories Memory-Map (continued)

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
System Modules Control Registers and Memories						
VIM RAM	PPCS2	0xFFF8_2000	0xFFF8_2FFF	4KB	1KB	Wrap around for accesses to unimplemented address offsets lower than 0x3FF. Accesses beyond 0x3FF will be ignored.
Flash Wrapper	PPCS7	0xFFF8_7000	0xFFF8_7FFF	4KB	4KB	Abort
eFuse Farm Controller	PPCS12	0xFFF8_C000	0xFFF8_CFFF	4KB	4KB	Abort
PCR registers	PPS0	0xFFFF_E000	0xFFFF_E0FF	256B	256B	Reads return zeros, writes have no effect
System Module - Frame 2	PPS0	0xFFFF_E100	0xFFFF_E1FF	256B	256B	Reads return zeros, writes have no effect
PBIST	PPS1	0xFFFF_E400	0xFFFF_E5FF	512B	512B	Reads return zeros, writes have no effect
STC	PPS1	0xFFFF_E600	0xFFFF_E6FF	256B	256B	Reads return zeros, writes have no effect
IOMM Multiplexing Control Module	PPS2	0xFFFF_EA00	0xFFFF_EBFF	512B	512B	Generates address error interrupt, if enabled
DCC	PPS3	0xFFFF_EC00	0xFFFF_ECFF	256B	256B	Reads return zeros, writes have no effect
ESM	PPS5	0xFFFF_F500	0xFFFF_F5FF	256B	256B	Reads return zeros, writes have no effect
CCM-R4	PPS5	0xFFFF_F600	0xFFFF_F6FF	256B	256B	Reads return zeros, writes have no effect
RAM ECC even	PPS6	0xFFFF_F800	0xFFFF_F8FF	256B	256B	Reads return zeros, writes have no effect
RAM ECC odd	PPS6	0xFFFF_F900	0xFFFF_F9FF	256B	256B	Reads return zeros, writes have no effect
RTI + DWWD	PPS7	0xFFFF_FC00	0xFFFF_FCFF	256B	256B	Reads return zeros, writes have no effect
VIM Parity	PPS7	0xFFFF_FD00	0xFFFF_FDFF	256B	256B	Reads return zeros, writes have no effect
VIM	PPS7	0xFFFF_FE00	0xFFFF_FEFF	256B	256B	Reads return zeros, writes have no effect
System Module - Frame 1	PPS7	0xFFFF_FF00	0xFFFF_FFFF	256B	256B	Reads return zeros, writes have no effect

2.2.3 Flash on Microcontrollers

The TMS570LS0232 microcontroller supports up to 128KB of flash for use as program memory. The microcontrollers also support a separate 16KB flash bank for use as emulated EEPROM.

2.2.3.1 Flash Bank Sectoring Configuration

The bank is divided into multiple sectors. A flash sector is the smallest region in the flash bank that must be erased. The sectoring configuration of each flash bank is shown in Table 2-4.

Refer to the device datasheet for electrical and timing specifications related to the flash module.

NOTE: The memory region from 0x0002_0000 to 0x0003_FFFF is not inherently protected against creating nERROR pin toggles from speculative fetches or from run away code. An MPU region must be used to limit accesses to only the first 128KB of ATCM space (0x0000_0000 to 0x0001_FFFF). Also, Flash API version 02.01.01 or later should be used to program or erase the flash of this device.

Table 2-4. Flash Memory Banks and Sectors (1)(2)(3)

Memory Arrays (or Banks)	Block No.	Sector No.	Segment	Low Address	High Address
BANK0 (384 KBytes)	0	0	8K Bytes	0x0000_0000	0x0000_1FFF
		1	8K Bytes	0x0000_2000	0x0000_3FFF
		2	8K Bytes	0x0000_4000	0x0000_5FFF
		3	8K Bytes	0x0000_6000	0x0000_7FFF
		4	8K Bytes	0x0000_8000	0x0000_9FFF
		5	8K Bytes	0x0000_A000	0x0000_BFFF
		6	8K Bytes	0x0000_C000	0x0000_DFFF
		7	8K Bytes	0x0000_E000	0x0000_FFFF
		8	8K Bytes	0x0001_0000	0x0001_1FFF
		9	8K Bytes	0x0001_2000	0x0001_3FFF
		10	8K Bytes	0x0001_4000	0x0001_5FFF
		11	8K Bytes	0x0001_6000	0x0001_7FFF
		12	32K Bytes	0x0001_8000	0x0001_FFFF
BANK7 (16 KBytes) for EEPROM emulation	0	0	4K Bytes	0xF020_0000	0xF020_0FFF
	1	1	4K Bytes	0xF020_1000	0xF020_1FFF
	2	2	4K Bytes	0xF020_2000	0xF020_2FFF
	3	3	4K Bytes	0xF020_3000	0xF020_3FFF

(1) The Flash banks are 144-bit wide bank with ECC support.
(2) The flash bank7 is a FLEE bank and can be programmed while executing code from flash bank0.
(3) Code execution is not allowed from flash bank7.

2.2.3.2 ECC Protection for Flash Accesses

The TMS570LS0232 microcontroller protects all accesses to the on-chip flash memory by dedicated Single-Bit Error Correction Double-Bit Error Detection (SECDED) logic.

The access to the program memory – flash bank 0, is protected by SECDED logic implemented inside the ARM Cortex-R4 CPU. Accesses to the EEPROM emulation flash bank (bank 7) are protected by dedicated SECDED logic inside the digital interface to the flash banks.

Both the SECDED logic implementations use Error Correction Codes (ECC) for correcting single-bit errors and for detecting multiple-bit errors in the values read from the flash arrays. There is an 8-bit ECC for every 64 bits of data. The ECC for the flash memory contents needs to be calculated by an external tool such as nowECC. The ECC can then be programmed into the flash array along with the actual application code.

The ECC for the flash array is stored in the flash itself, and is mapped to a region starting at 0xF0400000 for the main flash bank 0, and to a region starting at 0xF0100000 for the EEPROM emulation flash bank 7.

NOTE: ECC Protection Not Enabled By Default

The SECDED logic inside the CPU is not enabled by default and must be enabled by the application.

Example 2-1. Code Example for Enabling ECC Protection for Main Flash Accesses

```
MRC p15, #0, r1, c1, c0, #1
ORR r1, r1, #0x02000000       ;Enable ECC checking for ATCM
DMB
MCR p15, #0, r1, c1, c0, #1
```

The ECC protection for accesses to the EEPROM emulation flash bank can be enabled by writing Ah to the EDACEN field of the flash module's Error Correction Control Register 1 (FEDACCTRL1). See Chapter 4 for more details.

When the CPU detects an ECC single-, or double-bit error on a read from the flash memory, it signals this on a dedicated "*Event*" bus. This event bus signaling is also not enabled by default and must be enabled by the application.

Example 2-2. Code Example for Enabling the CPU Event Signaling

```
MRC p15,#0,r1,c9,c12,#0       ;Enabling Event monitor states
ORR r1, r1, #0x00000010
MCR p15,#0,r1,c9,c12,#0       ;Set 4th bit ('X') of PMNC register
MRC p15,#0,r1,c9,c12,#0
```

The digital logic that interfaces the ARM Cortex-R4 CPU to the flash banks captures the ECC error events signaled by the CPU, and in turn generates error signals that are input to the central Error Signaling Module (ESM) provided ECC protection is enabled within the digital logic by writing Ah to the EDACEN field of the flash module's Error Correction Control Register 1 (FEDACCTRL1).

NOTE: The state of the ECC bits for the memory region from 0x0002_0000 to 0x0003_FFFF is indeterminate and cannot be updated. This means, that accesses to this region of memory may or may not generate an ECC error (either single-bit correction or double-bit detection). To assure deterministic behavior, an MPU region must be used to limit accesses to only the first 128KB of ATCM space (0x0000_0000 to 0x0001_FFFF). Also, Flash API version 02.01.01 or later should be used to program or erase the flash of this device.

2.2.4 On-Chip SRAM

Several SRAM modules are implemented on the device to support the functionality of the modules included.

Reads from the CPU data RAM are protected by ECC calculated inside the CPU. Reads from all other memories are protected by configurable odd or even parity that is evaluated in parallel with the actual read.

The TMS570LS0232 microcontrollers are targeted towards safety-critical applications, and it is critical for any failures in the on-chip SRAM modules to be identified before these modules are used for safety-critical functions. These microcontrollers support a Programmable Built-In Self-Test (PBIST) mechanism that is used to test each on-chip SRAM module for faults. The PBIST is usually run on device start-up as it is a destructive test and all contents of the tested SRAM module are overwritten during the test.

The microcontrollers also support a hardware-based auto-initialization of on-chip SRAM modules. This process also takes into account the read protection scheme implemented for each SRAM module – ECC or parity.

TI recommends that the PBIST routines be executed on the SRAM modules prior to the auto-initialization. The following section describe these two processes.

2.2.4.1 PBIST RAM Grouping and Algorithm Mapping For On-Chip SRAM Modules

Table 2-5 shows the groupings of the various on-chip memories for PBIST. It also lists the memory types and their assigned RAM Group Select (RGS) and Return Data Select (RDS). See Chapter 6 for more details on the usage of the RGS and RDS information.

Table 2-6 maps the different algorithms supported in application mode for the RAM groups. The table also lists the background pattern options available for each algorithm.

NOTE: Recommended Memory Test Algorithm

March13 is the recommended algorithm for the memory self-test.

For PBIST ROM_CLK = HCLK/2.

Table 2-5. PBIST Memory Grouping

Memory	RAM Group#	Memory Type	RGS	RDS
PBIST_ROM	1	ROM	1	0
STC_ROM	2	ROM	2	0
DCAN1	3	Dual-port	3	0 .. 5
DCAN2	4	Dual-port	4	0 .. 5
ESRAM1	6	Single-port	6	0/1 .. 4
MIBSPI1	7	Dual-port	7	0 .. 3
VIM	10	Dual-port	8	0 .. 1
MIBADC	11	Dual-port	9	0
N2HET	13	Dual-port	11	0 .. 11
HET TU	14	Dual-port	12	0 .. 5

Table 2-6. PBIST Algorithm Mapping

Sr. No.	ALGO Register Value	Algorithm	Memories Under Test	Available Background Patterns	Valid RAM Groups	Valid RINFO Register Value
1	0x00000001	triple_read_slow_read	ROM		1,2	0x00000003
2	0x00000002	triple_read_fast_read	ROM		1,2	0x00000003
3	0x00000004	march13n	Dual-port	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3	3,4,7,10,11,13,14	0x0000364C
4	0x00000008	march13n	Single-port	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3	6,29	0x10000020
5	0x00000010	down1A_red	Dual-port	0xFFFFFFFF, 0xAAAAAAAA	3,4,7,10,11,13,14	0x0000364C
6	0x00000020	down1A_red	Single-port	0xFFFFFFFF, 0xAAAAAAAA	6,29	0x10000020
7	0x00000040	mapcolumn	Dual-port	0xFFFFFFFF, 0x00000000	3,4,7,10,11,13,14	0x0000364C
8	0x00000080	mapcolumn	Single-port	0xFFFFFFFF, 0x00000000	6,29	0x10000020
9	0x00000100	precharge	Dual-port	0xFFFFFFFF, 0x00000000	3,4,7,10,11,13,14	0x0000364C
10	0x00000200	precharge	Single-port	0xFFFFFFFF, 0x00000000	6,29	0x10000020
11	0x00000400	dtxn2	Dual-port	0xFFFFFFFF, 0x00000000	3,4,7,10,11,13,14	0x0000364C
12	0x00000800	dtxn2	Single-port	0xFFFFFFFF, 0x00000000	6,29	0x10000020
13	0x00001000	pmos_open	Dual-port	0xFFFFFFFF, 0x00000000	3,4,7,10,11,13,14	0x0000364C
14	0x00002000	pmos_open	Single-port	0xFFFFFFFF, 0x00000000	6,29	0x10000020
15	0x00004000	pmos_open_slice1	Dual-port	0xFFFFFFFF, 0x00000000	10	0x00000200
16	0x00008000	pmos_open_slice2	Dual-port	0xFFFFFFFF, 0x00000000	10	0x00000200
17	0x00010000	flip10	Dual-port	0xFFFFFFFF	3,4,7,10,11,13,14	0x0000364C
18	0x00020000	flip10	Single-port	0xFFFFFFFF	6,29	0x10000020
19	0x00040000	iddq	Dual-port	0x00000000	3,4,7,10,11,13,14	0x0000364C
20	0x00080000	iddq	Single-port	0x00000000	6	0x00000020
21	0x00100000	retention	Dual-port	0x00000000	3,4,7,10,11,13,14	0x0000364C
22	0x00200000	retention	Single-port	0x00000000	6	0x00000020
23	0x00400000	iddq	Dual-port	0xFFFFFFFF	3,4,7,10,11,13,14	0x0000364C
24	0x00800000	iddq	Single-port	0xFFFFFFFF	6	0x00000020
25	0x01000000	retention	Dual-port	0xFFFFFFFF	3,4,7,10,11,13,14	0x0000364C
26	0x02000000	retention	Single-port	0xFFFFFFFF	6	0x00000020
27	0x04000000	iddqrowstripe	Dual-port	0x00000000	3,4,7,10,11,13,14	0x0000364C
28	0x08000000	iddqrowstripe	Single-port	0x00000000	6	0x00000020
29	0x10000000	iddqrowstripe	Dual-port	0xFFFFFFFF	3,4,7,10,11,13,14	0x0000364C
30	0x20000000	iddqrowstripe	Single-port	0xFFFFFFFF	6	0x00000020
31	0x40000000	powerup_invpowerup	Dual-port	0xAAAAAAAA	33,34,36,39,40,42,43	0x000006CB
32	0x80000000	powerup_invpowerup	Single-port	0xAAAAAAAA	52	0x00080000

2.2.4.2 Auto-Initialization of On-Chip SRAM Modules

The device system provides the capability to perform a hardware initialization on most memories on the system bus and on the peripheral bus.

The intent of having the hardware initialization is to program the memory arrays with error detection capability to a known state based on their error detection scheme – odd/even parity or ECC. For example, the contents of the CPU data RAM after power-on reset is unknown. A hardware auto-initialization can be started so that there is no ECC error.

NOTE: Effect of ECC or Parity on Memory Auto-Initialization

The ECC or parity should be enabled on the RAMs before hardware auto-initialization starts if parity or ECC is being used.

Auto-Initialization Sequence:

1. Enable the global hardware memory initialization key by programming 0xA into MINITGCR[3:0], the Memory Initialization Key field (MINITGENA) of the Memory Hardware Initialization Global Control Register (MINITGCR) register.
2. Select the module on which the memory hardware initialization has to be performed by programming the appropriate value into the MSINENA(31–0) bits in the MSINENA register. See Table 2-7.
3. If the global auto-initialization scheme is enabled, the corresponding module will initialize its memories based on its memory error checking scheme (even parity or odd parity or ECC).
4. When the memory initialization is complete, the module will signal “memory initialization done”, which sets the corresponding bit in the system module MIDONE field of the MINISTAT register to indicate the completion of its memory initialization.
5. When the memory hardware initialization completes for all modules, (indicated by each module’s MIDONE bit being set), the memory hardware initialization done bit (MINIDONE) is set in the MSTCGSTAT register.

Figure 2-3. Hardware Memory Initialization Protocol

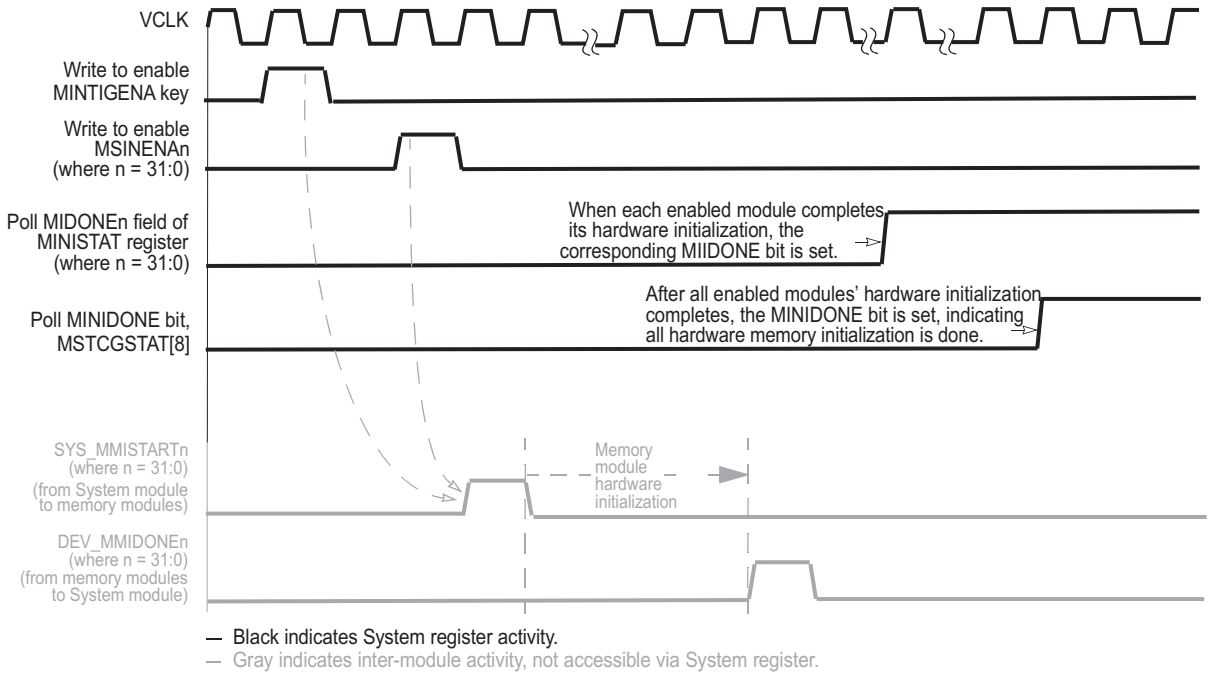


Table 2-7. Memory Initialization Select Mapping [(1)(2)]

Memory	Address Range		MSINENA Register Bit #
	Start	End	
RAM	0x08000000	0x0800FFFF	0
MIBSPI1 RAM	0xFF0E0000	0xFF0FFFFF	7[(3)]
DCAN2 RAM	0xFF1C0000	0xFF1DFFFF	6
DCAN1 RAM	0xFF1E0000	0xFF1FFFFF	5
MIBADC RAM	0xFF3E0000	0xFF3FFFFF	8
N2HET RAM	0xFF460000	0xFF47FFFF	3
HET TU RAM	0xFF4E0000	0xFF4FFFFF	4
VIM RAM	0xFFF82000	0xFFF82FFF	2

(1) If ECC protection is enabled for the CPU data RAM, then the auto-initialization process also initializes the corresponding ECC space.

(2) If parity protection is enabled for the peripheral SRAM modules, then the parity bits will also be initialized along with the SRAM modules.

(3) The MibSPI module performs an initialization of the transmit and receive RAMs as soon as the multi-buffered mode is enabled. This is independent of whether the application has already initialized these RAMs using the auto-initialization method or not. The MibSPI module need to be released from reset by writing 1 to the SPIGCR0 register before starting auto-initialization on its RAMs.

2.3 Exceptions

An “Exception” is an event that makes the processor temporarily halt the normal flow of program execution, for example, to service an interrupt from a peripheral. Before attempting to handle an exception, the processor preserves the critical parts of the current processor state so that the original program can resume when the handler routine has finished.

The following sections describe three exceptions – Reset, Abort and the System Software Interrupts.

For complete details on all exceptions, refer to the ARM® Cortex®-R4 Technical Reference Manual.

2.3.1 Resets

The TMS570LS0232 microcontroller can be reset by either of the conditions described in Table 2-8. Each reset condition is indicated in the System Exception Status Register (SYSESR).

The device nRST terminal is an I/O. It can be driven low by an external circuit to force a warm reset on the microcontroller. This terminal will be driven low as an output for a minimum of 32 peripheral clock (VCLK) cycles for any device system reset condition. As a result the EXTRST bit in the SYSESR register, SYSESR[3], gets set for all reset conditions listed in Table 2-8. The nRST is driven low as an output for a longer duration during device power-up or whenever the power-on reset (nPORRST) is driven low externally. Refer the device datasheet for the electrical and timing specifications for the nRST.

Table 2-8. Causes of Resets

Condition	Description
Driving nPORRST pin low externally	Cold reset, or power-on reset. This reset signal is typically driven by an external voltage supervisor. This reset is flagged by the PORST bit in the SYSESR register, SYSESR[15].
Voltage Monitor reset	The microcontroller has an embedded voltage monitor that generates a power-on reset when the core voltage gets out of a valid range, or when the I/O voltage falls below a threshold. This reset is also flagged by the PORST bit in the SYSESR register, SYSESR[15]. **Note:** The voltage monitor is not an alternative for an external voltage supervisor.
Driving nRST pin low externally	Warm reset. This reset input is typically used in a system with multiple ICs and which requires that the microcontroller also gets reset whenever the other IC detects a fault condition. This reset is flagged by the EXTRST bit in the SYSESR, register SYSESR[3].
Oscillator failure	This reset is generated by the system module when the clock monitor detects an oscillator fail condition. Whether or not a reset is generated is also dictated by a register in the system module. This reset is flagged by the OSCRST bit in the SYSESR register, SYSESR[14].
Software reset	This reset is generated by the application software writing a 1 to bit 15 of System Exception Control Register (SYSECR) or a 0 to bit 14 of SYSECR. It is typically used by a bootloader type of code that uses a software reset to allow the code execution to branch to the application code once it is programmed into the program memory. This reset is flagged by the SWRST bit in the SYSESR register, SYSESR[4].
CPU reset	This reset is generated by the CPU self-test controller (LBIST) or by changing the memory protection (MMU/MPU) configuration in the CPURSTCR register. This reset is flagged by the CPURST bit in the SYSESR register, SYSESR[5].
Debug reset	The ICEPICK logic implemented on the microcontroller allows a system reset to be generated via the debug logic. This reset is flagged by the WDRST bit in the SYSESR register, SYSESR[13].
Watchdog reset	This reset is generated by the digital windowed watchdog (DWWD) module on the microcontroller. The DWWD can generate a reset whenever the watchdog service window is violated. This reset is flagged by the WDRST bit in the SYSESR register, SYSESR[13].

2.3.2 Aborts

When the ARM Cortex-R4 processor's memory system cannot complete a memory access successfully, an abort is generated. An error occurring on an instruction fetch generates a ***prefetch abort***. Errors occurring on data accesses generate ***data aborts***. Aborts are also categorized as being either ***precise*** or ***imprecise***.

2.3.2.1 Prefetch Aborts

When a Prefetch Abort (PABT) occurs, the processor marks the prefetched instruction as invalid, but does not take the exception until the instruction is to be executed. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the abort does not take place.

All prefetch aborts are precise aborts.

2.3.2.2 Data Aborts

An error occurring on a data memory access can generate a data abort. If the instruction generating the memory access is not executed, for example, because it fails its condition codes, or is interrupted, the data abort does not take place.

A Data Abort (DABT) can be either precise or imprecise, depending on the type of fault that caused it.

2.3.2.3 Precise Aborts

A precise abort, also known as a synchronous abort, is one for which the exception is guaranteed to be taken on the instruction that generated the aborting memory access. The abort handler can use the value in the Link Register (r14_abt) to determine which instruction generated the abort, and the value in the Saved Program Status Register (SPSR_abt) to determine the state of the processor when the abort occurred.

2.3.2.4 Imprecise Aborts

An imprecise abort, also known as an asynchronous abort, is one for which the exception is taken on a later instruction to the instruction that generated the aborting memory access. The abort handler cannot determine which instruction generated the abort, or the state of the processor when the abort occurred. Therefore, imprecise aborts are normally fatal.

Imprecise aborts can be generated by store instructions to normal-type or device-type memory. When the store instruction is committed, the data is normally written into a buffer that holds the data until the memory system has sufficient bandwidth to perform the write access. This gives read accesses higher priority. The write data can be held in the buffer for a long period, during which many other instructions can complete. If an error occurs when the write is finally performed, this generates an imprecise abort.

The TMS570LS0232 microcontroller architecture applies techniques at the system level to mitigate the impact of imprecise aborts. System level adoption of write status sidebands to the data path allow bus masters to comprehend imprecise aborts, turning them into precise aborts. In cases where this approach is not feasible, buffering bridges or other sources of imprecision may build a FIFO of current transactions such that an imprecise abort may be registered at the point of imprecision for later analysis.

Masking Of Imprecise Aborts:

The nature of imprecise aborts means that they can occur while the processor is handling a different abort. If an imprecise abort generates a new exception in such a situation, the banked link register (R14_abt) and the Saved Processor Status Register (SPSR_abt) values are overwritten. If this occurs before the data is pushed to the stack in memory, the state information about the first abort is lost. To prevent this from happening, the Current Processor Status Register (CPSR) contains a mask bit to indicate that an imprecise abort cannot be accepted, the A-bit. When the A-bit is set, any imprecise abort that occurs is held pending by the processor until the A-bit is cleared, when the exception is actually taken. The A-bit is automatically set when abort, IRQ or FIQ exceptions are taken, and on reset. The application must only clear the A-bit in an abort handler after the state information has either been stacked to memory, or is no longer required.

NOTE: Default Behavior for Imprecise Aborts

The A-bit in the CPSR is set by default. This means that no imprecise abort exception will occur. The application must enable imprecise abort exception generation by clearing the A-bit of the CPSR.

2.3.2.5 Conditions That Generate Aborts

An Abort is generated under the following conditions on the TMS570LS0232 microcontrollers.

- Access to an illegal address (a nonimplemented address)
- Access to a protected address (protection violation)
- Parity / ECC / Time-out Error on a valid access

Illegal Addresses:

The illegal addresses and the responses to an access to these addresses are defined in Table 2-3.

Addresses Protected By MPU:

Memory access permissions can be configured via the ARM Cortex-R4 processor's Memory Protection Unit (MPU). For more details on the MPU configuration, refer to the ARM® Cortex®-R4 Technical Reference Manual.

A memory access violation is logged as a permission fault in the CPU's fault status register and the virtual address of the access is logged into the CPU's fault address register.

Protection of Peripheral Register and Memory Frames:

Accesses to the peripheral register and memory frames can be protected either by configuring the MPU or by configuring the Peripheral Central Resource (PCR) controller registers.

The PCR module PPROTSETx registers contain one bit per peripheral select quadrant. These bits define the access permissions to the peripheral register frames. If the CPU attempts to write to a peripheral register for which it does not have the correct permissions, a protection violation is detected and an Abort occurs.

Some modules also enforce register updates to only be allowed when the CPU is in a privileged mode of operation. If the CPU writes to these registers in user mode, the writes are ignored.

The PCR module PMPROTSETx registers contain one bit per peripheral memory frame. These bits define the access permissions to the peripheral memory frames. If the CPU attempts to write to a peripheral memory for which it does not have the correct permissions, a protection violation is detected and an Abort occurs.

NOTE: No Access Protection for Reads

The PCR PPROTSETx and PMPROTSETx registers protect the peripheral registers and memories against illegal writes by the CPU. The CPU can read from the peripheral registers and memories in both user and privileged modes.

2.3.3 System Software Interrupts

The system module provides the capability of generating up to four software interrupts. A software interrupt is generated by writing the correct key value to either of the four System Software Interrupt Registers (SSIRx). The SSI registers also allow the application to provide a label for that software interrupt. This label is an 8-bit value that can then be used by the interrupt service routine to perform the required task based on the value provided. The source of the system software interrupt is reflected in the system software interrupt vector (SSIVEC) register.

2.4 Clocks

This section describes the clocking structure of the TMS570LS0232 microcontrollers.

2.4.1 Clock Sources

The devices support up to 5 clock sources. These are listed in Table 2-9. The electrical specifications as well as timing requirements for each of the clock sources are specified in the device datasheet.

Table 2-9. Clock Sources

Clock Source #	Clock Source Name	Description
0	OSCIN	Main oscillator. This is the primary clock for the microcontroller and is the only clock that is input to the phase-locked loops. The oscillator frequency must be between 5 and 20 MHZ.
1	PLL1	This is the output of the main PLL. The PLL is capable of modulating its output frequency in a controlled manner to reduce the radiated emissions.
2	Reserved	This clock source is not available and must not be enabled or used as source for any clock domain.
3	EXTCLKIN	External clock input. A square wave input can be applied to this device input and used as a clock source inside the device. Use of EXTCLKIN requires selecting the EXTCLKIN pin function in the PINMMR2 register.
4	LF LPO (Low-Frequency LPO) CLK80K	This is the low-frequency output of the internal reference oscillator. This is typically an 80 KHz signal (CLK80K) that is used by the real-time interrupt module for generating periodic interrupts to wake up from a low power mode.
5	HF LPO (High-Frequency LPO) CLK10M	This is the high-frequency output of the internal reference oscillator. This is typically a 10 MHZ signal (CLK10M) that is used by the clock monitor module as a reference clock to monitor the main oscillator frequency.
6	Reserved	This clock source is not available and must not be enabled or used as source for any clock domain.
7	Reserved	This clock source is not available and must not be enabled or used as source for any clock domain.

2.4.1.1 Enabling / Disabling Clock Sources

Each clock source can be independently enabled or disabled using the set of Clock Source Disable registers – CSDIS, CSDISSET and CSDISCLR.

Each bit in these registers corresponds to the clock source number indicated in Table 2-9. For example, setting bit 1 in the CSDIS or CSDISSET registers disables the PLL.

NOTE: Disabling the Main Oscillator or HF LPO

By default, the clock monitoring circuit is enabled and checks for the main oscillator frequency to be within a certain range using the HF LPO as a reference. If the main oscillator and/or the HF LPO are disabled with the clock monitoring still enabled, the clock monitor will indicate an oscillator fault. The clock monitoring must be disabled before disabling the main oscillator or the HF LPO clock source(s).

The clock source is only disabled once there is no active clock domain that is using that clock source. Also, see Chapter 9 for more information on enabling/disabling the oscillator and PLL.

On the TMS570LS0232 microcontrollers, the clock sources 0, 4, and 5 are enabled by default.

2.4.1.2 Checking for Valid Clock Sources

The application can check whether a clock source is valid or not by checking the corresponding bit to be set in the Clock Source Valid Status (CSVSTAT) register. For example, the application can check if bit 1 in CSVSTAT is set before using the output of PLL as the source for any clock domain.

2.4.2 Clock Domains

The clocking on this device is divided into multiple clock domains for flexibility in control as well as clock source selection. There are 7 clock domains on this device. Each of these are described in Table 2-10.

Each of the control registers listed in Table 2-10 are defined in Section 2.5. The AC timing characteristics for each clock domain are specified in the device datasheet.

Table 2-10. Clock Domains

Clock Domain #	Clock Domain	Default Source	Source Selection Register	Special Considerations
0	GCLK	OSCIN	GHVSRC	• Always the same frequency as HCLK • In phase with HCLK • Is disabled separately from HCLK via the CDDISx registers bit 0 • Can be divided by 1 to 8 when running CPU self-test (LBIST) using the CLKDIV field of the STCCLKDIV register
0	GCLK2	OSCIN	GHVSRC	• Always the same frequency as GCLK • 2 cycles delayed from GCLK • Is disabled along with GCLK • Gets divided by the same divider setting as that for GCLK when running CPU self-test (LBIST)
1	HCLK	OSCIN	GHVSRC	• Is disabled via the CDDISx registers bit 1
2	VCLK	OSCIN	GHVSRC	• Divided down from HCLK • Can be HCLK/1 or HCLK/2 • Is disabled separately from HCLK via the CDDISx register bit 2 • Can be disabled separately for the eQEP module via the CDDISx register bit 9.
3	VCLK2	OSCIN	GHVSRC	• Divided down from HCLK • Can be HCLK/1 or HCLK/2 • Frequency must be an integer multiple of VCLK frequency • Is disabled separately from HCLK via the CDDISx registers bit 3
4	VCLKA1	VCLK	VCLKASRC	• Defaults to VCLK as the source • Frequency can be as fast as HCLK frequency • Is disabled via the CDDISx registers bit 4
6	RTICLK	VCLK	RCLKSRC	• Defaults to VCLK as the source • If a clock source other than VCLK is selected for RTICLK, then the RTICLK frequency must be less than or equal to VCLK/3 . The application can ensure this by programming the RTI1DIV field of the RCLKSRC register, if necessary • Is disabled via the CDDISx registers bit 6

2.4.2.1 Enabling / Disabling Clock Domains

Each clock domain can be independently enabled or disabled using the set of Clock Domain Disable registers – CDDIS, CDDISSET, and CDDISCLR.

Each bit in these registers corresponds to the clock domain number indicated in Table 2-10. For example, setting bit 1 in the CDDIS or CDDISSET registers disables the HCLK clock domain. The clock domain will be turned off only when every module that uses the HCLK domain gives the “permission” for HCLK to be turned off.

All clock domains are enabled by default, or upon a system reset, or whenever a wake up condition is detected.

2.4.2.2 Mapping Clock Sources to Clock Domains

Each clock domain needs to be mapped to a valid clock source. There are control registers that allow an application to choose the clock sources for each clock domain.

- ***Selecting clock source for GCLK, HCLK and VCLKx domains***

The CPU clock (GCLK), the system module clock (HCLK), and the peripheral bus clocks (VCLKx) all use the same clock source. This clock source is selected via the GHVSRC register. The default source for the GCLK, HCLK and VCLKx is the main oscillator. That is, after power up, the GCLK and HCLK are running at the OSCIN frequency, while the VCLKx frequency is the OSCIN frequency divided by 2.

- ***Selecting clock source for the VCLKA1 domain***

The clock source for the VCLKA1 domain is selected via the VCLKASRC register. The default source for the VCLKA1 domain is the VCLK.

- ***Selecting clock source for RTICLK domain***

The clock source for RTICLK domain is selected via the RCLKSRC register. The default source for the RTICLK domain is the VCLK.

NOTE: Selecting a clock source for RTICLK that is not VCLK

When the application chooses a clock source for RTICLK domain that is not VCLK, then the application must ensure that the resulting RTICLK frequency must be less than or equal to VCLK frequency divided by 3. The application can configure the RTI1DIV field of the RCLKSRC register for dividing the selected clock source frequency by 1, 2, 4 or 8 to meet this requirement.

2.4.3 Low-Power Modes

All clock domains are active in the normal operating mode. This is the default mode of operation. As described in Section 2.4.1.1 and Section 2.4.2.1, the application can choose to disable any particular clock source and domain that it does not plan to use. Also, the peripheral central resource controller (PCR) has control registers to enable / disable the peripheral clock (VCLK) for each peripheral select. This offers the application a large number of choices for enabling / disabling clock sources, or clock domains, or clocks to specific peripherals.

This section describes three particular low-power modes and their typical characteristics. They are not the only low-power modes configurable by the application, as just described.

Table 2-11. Typical Low-Power Modes

Mode Name	Active Clock Source(s)	Active Clock Domain(s)	Wake Up Options	Suggested Wake Up Clock Source(s)	Wake Up Time (wake up detected -to-CPU code execution start)
Doze	Main oscillator	RTICLK	RTI interrupt, GIO interrupt, CAN message, SCI message	Main oscillator	Flash pump sleep -> active transition time + Flash bank sleep -> standby transition time + Flash bank standby -> active transition time
Snooze	LF LPO	RTICLK	RTI interrupt, GIO interrupt, CAN message, SCI message	HF LPO	HF LPO warm start-up time + Flash pump sleep -> active transition time + Flash bank sleep -> standby transition time + Flash bank standby -> active transition time
Sleep	None	None	GIO interrupt, CAN message, SCI message	HF LPO	HF LPO warm start-up time + Flash pump sleep -> active transition time + Flash bank sleep -> standby transition time + Flash bank standby -> active transition time

2.4.3.1 Typical Software Sequence to Enter a Low-Power Mode

1. Program the flash banks and flash pump fall-back modes to be “sleep”.

   The flash pump transitions from active to sleep mode only after all the flash banks have switched from active to sleep mode. The flash banks start switching from active to sleep mode only after the banks are not accessed for at least a duration defined by the Active Grace Period (AGP) parameter configured for the banks. See Chapter 4 for more details.
2. Disable the clock sources that are not required to be kept active.

   A clock source does not get disabled until all clock domains using that clock source are disabled first, or are configured to use an alternate clock source.
3. Disable the clock domains that are not required to be kept active.

   A clock domain does not get disabled until all modules using that clock domain “give their permission” for that clock domain to be turned off.
4. Idle the Cortex-R4 core.

   The ARM Cortex-R4 CPU has internal power management logic, and requires a dedicated instruction to be used in order to enter a low power mode. This is the Wait For Interrupt (WFI) instruction. When a WFI instruction is executed, the Cortex-R4 core flushes its pipeline, flushes all write buffers, and completes all pending bus transactions. At this time the core indicates to the system that the clock to the core can be stopped. This indication is used by the Global Clock Module (GCM) to turn off the CPU clock domain (GCLK) if the CDDIS register bit 0 is set.

2.4.3.2 Special Considerations for Entry to Low-Power Modes

The bus masters can have ongoing transactions when the application wants to enter a low-power mode to turn off the clocks to those modules. This is not recommended as it could leave the device in an unpredictable state. See the individual module chapters for more information about the sequence to be followed to safely enter a low-power mode.

2.4.3.3 Selecting Clock Source Upon Wake Up

The domains for CPU clock (GCLK), the system clock (HCLK) and the peripheral clock (VCLKx) use the same clock source selected via the GHVSRC field of the GHVSRC register. The GHVSRC register also allows the application to choose the clock source after wake up via the GHVWAKE field.

When a wake up condition is detected, if the selected wake up clock source is not already active, the global clock module (GCM) will enable this selected clock source, wait for it to become valid, and then use it for the GCLK, HCLK and VCLKx domains. The other clock domains VCLKAx and RTICLK retain the configuration for their clock source selection registers – VCLKASRC, VCLKACON1 and RCLKSRC.

2.4.4 Clock Test Mode

The TMS570LS0232 microcontrollers support a test mode which allows a user to bring out several different clock sources and clock domains on to the ECLK terminal. This is very useful information for debug purposes. Each clock source also has a corresponding clock source valid status flag in the Clock Source Valid Status (CSVSTAT) register. The clock source valid status flags can also be brought out on to the N2HET[2] terminal in this clock test mode.

The clock test mode is controlled by the CLKTEST register (Section 2.5.1.32) in the system module register frame.

The clock test mode is enabled by writing 0x5 to the CLK_TEST_EN field.

The signal to be brought out on to the ECLK terminal is defined by the SEL_ECP_PIN field and the signal to be brought out on to the N2HET[2] terminal is defined by the SEL_N2HET_PIN field.

These selections are defined in Table 2-12.

Table 2-12. Clock Test Mode Options

SEL_ECP_PIN	Signal on ECLK		SEL_N2HET_PIN	Signal on N2HET[2]
0000	Oscillator		0000	Oscillator Valid status
0001	PLL1 free-running clock output (PLLCLK)		0001	PLL1 Valid status
0010	Reserved		0010	Reserved
0011	Reserved		0011	Reserved
0100	CLK80K		0100	Reserved
0101	CLK10M		0101	CLK10M Valid status
0110	Reserved		0110	Reserved
0111	Reserved		0111	Reserved
1000	GCLK		1000	CLK80K Valid status
1001	RTI Base		1001	Oscillator Valid status
1010	Reserved		1010	Oscillator Valid status
1011	VCLKA1		1011	Oscillator Valid status
1100	Reserved		1100	Oscillator Valid status
1101	Reserved		1101	Oscillator Valid status
1110	Reserved		1110	Oscillator Valid status
1111	Flash HD Pump Oscillator		1111	Oscillator Valid status

2.4.5 Safety Considerations for Clocks

The TMS570LS0232 microcontrollers are targeted for use in several safety-critical applications. The following sections describe the internal or external monitoring mechanisms that detect and signal clock source failures.

2.4.5.1 Oscillator Monitor

The oscillator clock frequency is monitored by a dedicated circuitry called CLKDET using the HF LPO as the reference clock. The CLKDET flags an oscillator fail condition whenever the OSCIN frequency falls outside of a range which is defined by the HF LPO frequency. Please refer to the device datasheet for specific frequency values.

The valid OSCIN range is defined as a minimum of $f_{(HF\ LPO)}$ / 4 to a maximum of $f_{(HF\ LPO)}$ × 4.

The application can select the device response to an oscillator fail indication. See Chapter 9 for more details on the oscillator monitoring and the system response choices.

2.4.5.2 PLL Slip Detector

The PLL macro implemented on the microcontroller has an embedded slip detection circuit. A PLL slip is detected by the slip detector under the following conditions:

1. Reference cycle slip, RFSLIP — the output clock is running *too fast* relative to the reference clock
2. Feedback cycle slip, FBSLIP — the output clock is running *too slow* relative to the reference clock

The device also includes optional filters that can be enabled before a slip indication from the PLL is actually logged in the system module Global Status Register (GLBSTAT). Also, once a PLL slip condition is logged in the system module global status register, the application can choose the device's response to the slip indication. See Chapter 9 for more details on PLL slip and the system response choices.

2.4.5.3 External Clock Monitor

The microcontrollers support a terminal called ECLK – External Clock, which is used to output a slow frequency which is divided down from the device system clock frequency. An external circuit can monitor the ECLK frequency in order to check that the device is operating at the correct frequency.

The frequency of the signal output on the ECLK pin can be divided down by 1 to 65536 from the peripheral clock (VCLK) frequency using the External Clock Prescaler Control Register (ECPCNTL). The actual clock output on ECLK is enabled by setting the ECP CLK FUN bit of the SYSPC1 control register.

2.4.5.4 Dual-Clock Comparator

The microcontroller includes a dual-clock comparator module. This module includes two down counters which independently count from two separate seed values at the rate of two independent clock frequencies. One of the clock inputs is a reference clock input, selectable between the main oscillator or the HF LPO in functional mode. The second clock input is selectable from among a set of defined signals. This mechanism can be used to use a known-good clock to measure the frequency of another clock.

The DCC module has two counters. Table 2-13 and Table 2-14 show the options for selecting the clock sources for both counters on the DCC module.

CNT0 CLKSRC is a control field in the DCCCNT0CLKSRC register. KEY and CNT1 CLKSRC are control fields in the DCCCNT1CLKSRC register.

As can be seen, the main oscillator (OSCIN) can be used for counter 0 as a “known-good” reference clock. The clock for counter 1 can be selected from among 5 options. See Chapter 10 for more details on the DCC usage.

Table 2-13. Clock Source Selection for DCC1 Counter0

CNT0 CLKSRC	Clock Name	Clock Description
5h	HF LPO	High-Frequency Output of Internal Low-Power Oscillator
Ah	TCK	Test Clock
All other values	OSCIN	Main oscillator input

Table 2-14. Clock Source Selection for DCC1 Counter1

KEY	CNT1 CLKSRC	Clock Name	Clock Description
Ah	0	PLL Output	Output from PLL
	1h	Reserved	Reserved
	2h	LF LPO	Low-Frequency output of internal Low-Power Oscillator
	3h	HF LPO	High-Frequency output of internal Low-Power Oscillator
	4h	Reserved	Reserved
	5h	EXTCLKIN	External clock input source #1
	6h	Reserved	Reserved
	7h	Reserved	Reserved
	8h–Fh	VCLK	Peripheral bus clock
All other values	Don’t care	N2HET[31]	Signal output on the N2HET[31] signal by the N2HET module, either by the N2HET program execution, or by the application toggling the N2HET[31] pin as general-purpose output.

2.5 System and Peripheral Control Registers

The following sections describe the system and peripheral control registers of the TMS570LS0232 microcontroller.

2.5.1 Primary System Control Registers (SYS)

This section describes the SYSTEM registers. These registers are broken up into two separate frames. The start address of the primary system module frame is FFFF FF00h. The start address of the secondary system module frame is FFFF E100h. The registers support 32-, 16-, and 8-bit writes. The offset is relative to the system module frame start address.

Table 2-15 contains a summary of the primary system control registers.

Table 2-15. Primary System Control Registers

Offset	Acronym	Register Description	Section
00h	SYSPC1	SYS Pin Control Register 1	Section 2.5.1.1
04h	SYSPC2	SYS Pin Control Register 2	Section 2.5.1.2
08h	SYSPC3	SYS Pin Control Register 3	Section 2.5.1.3
0Ch	SYSPC4	SYS Pin Control Register 4	Section 2.5.1.4
10h	SYSPC5	SYS Pin Control Register 5	Section 2.5.1.5
14h	SYSPC6	SYS Pin Control Register 6	Section 2.5.1.6
18h	SYSPC7	SYS Pin Control Register 7	Section 2.5.1.7
1Ch	SYSPC8	SYS Pin Control Register 8	Section 2.5.1.8
20h	SYSPC9	SYS Pin Control Register 9	Section 2.5.1.9
30h	CSDIS	Clock Source Disable Register	Section 2.5.1.10
34h	CSDISSET	Clock Source Disable Set Register	Section 2.5.1.11
38h	CSDISCLR	Clock Source Disable Clear Register	Section 2.5.1.12
3Ch	CDDIS	Clock Domain Disable Register	Section 2.5.1.13
40h	CDDISSET	Clock Domain Disable Set Register	Section 2.5.1.14
44h	CDDISCLR	Clock Domain Disable Clear Register	Section 2.5.1.15
48h	GHVSRC	GCLK, HCLK, VCLK, and VCLK2 Source Register	Section 2.5.1.16
4Ch	VCLKASRC	Peripheral Asynchronous Clock Source Register	Section 2.5.1.17
50h	RCLKSRC	RTI Clock Source Register	Section 2.5.1.18
54h	CSVSTAT	Clock Source Valid Status Register	Section 2.5.1.19
58h	MSTGCR	Memory Self-Test Global Control Register	Section 2.5.1.20
5Ch	MINITGCR	Memory Hardware Initialization Global Control Register	Section 2.5.1.21
60h	MSINENA	Memory Self-Test/Initialization Enable Register	Section 2.5.1.22
64h	MSTFAIL	Memory Self-Test Fail Status Register	Section 2.5.1.23
68h	MSTCGSTAT	MSTC Global Status Register	Section 2.5.1.24
6Ch	MINISTAT	Memory Hardware Initialization Status Register	Section 2.5.1.25
70h	PLLCTL1	PLL Control Register 1	Section 2.5.1.26
74h	PLLCTL2	PLL Control Register 2	Section 2.5.1.27
78h	SYSPC10	SYS Pin Control Register 10	Section 2.5.1.28
7Ch	DIEIDL	Die Identification Register, Lower Word	Section 2.5.1.29
80h	DIEIDH	Die Identification Register, Upper Word	Section 2.5.1.30
88h	LPOMONCTL	LPO/Clock Monitor Control Register	Section 2.5.1.31
8Ch	CLKTEST	Clock Test Register	Section 2.5.1.32
90h	DFTCTRLREG	DFT Control Register	Section 2.5.1.33
94h	DFTCTRLREG2	DFT Control Register 2	Section 2.5.1.34
A0h	GPREG1	General Purpose Register	Section 2.5.1.35
A8h	IMPFASTS	Imprecise Fault Status Register	Section 2.5.1.36
ACh	IMPFTADD	Imprecise Fault Write Address Register	Section 2.5.1.37

Table 2-15. Primary System Control Registers (continued)

Offset	Acronym	Register Description	Section
B0h	SSIR1	System Software Interrupt Request 1 Register	Section 2.5.1.38
B4h	SSIR2	System Software Interrupt Request 2 Register	Section 2.5.1.39
B8h	SSIR3	System Software Interrupt Request 3 Register	Section 2.5.1.40
BCh	SSIR4	System Software Interrupt Request 4 Register	Section 2.5.1.41
C0h	RAMGCR	RAM Control Register	Section 2.5.1.42
C4h	BMMCR1	Bus Matrix Module Control Register 1	Section 2.5.1.43
CCh	CPURSTCR	CPU Reset Control Register	Section 2.5.1.44
D0h	CLKCNTL	Clock Control Register	Section 2.5.1.45
D4h	ECPCNTL	ECP Control Register	Section 2.5.1.46
DCh	DEVCR1	DEV Parity Control Register 1	Section 2.5.1.47
E0h	SYSECR	System Exception Control Register	Section 2.5.1.48
E4h	SYSESR	System Exception Status Register	Section 2.5.1.49
E8h	SYSTASR	System Test Abort Status Register	Section 2.5.1.50
ECh	GLBSTAT	Global Status Register	Section 2.5.1.51
F0h	DEVID	Device Identification Register	Section 2.5.1.52
F4h	SSIVEC	Software Interrupt Vector Register	Section 2.5.1.53
F8h	SSIF	System Software Interrupt Flag Register	Section 2.5.1.54

2.5.1.1 SYS Pin Control Register 1 (SYSPC1)

The SYSPC1 register, shown in Figure 2-4 and described in Table 2-16, controls the function of the ECLK pin.

Figure 2-4. SYS Pin Control Register 1 (SYSPC1) (offset = 00h)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKFUN
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-16. SYS Pin Control Register 1 (SYSPC1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKFUN		ECLK function. This bit changes the function of the ECLK pin.
		0	ECLK is in GIO mode.
		1	ECLK is in functional mode as a clock output.
			Note: Proper ECLK duty cycle is not assured until 1 ECLK cycle has elapsed after switching into functional mode.

2.5.1.2 SYS Pin Control Register 2 (SYSPC2)

The SYSPC2 register, shown in Figure 2-5 and described in Table 2-17, controls whether the pin is an input or an output when in GIO mode.

Figure 2-5. SYS Pin Control Register 2 (SYSPC2) (offset = 04h)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKDIR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-17. SYS Pin Control Register 2 (SYSPC2) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKDIR		ECLK data direction. This bit controls the direction of the ECLK pin when it is configured to be in GIO mode only.
		0	The ECLK pin is an input.
			Note: If the pin direction is set as an input, the output buffer is tristated.
		1	The ECLK pin is an output.
			Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register.

2.5.1.3 SYS Pin Control Register 3 (SYSPC3)

The SYSPC3 register, shown in Figure 2-6 and described in Table 2-18, displays the logic state of the ECLK pin when it is in GIO mode.

Figure 2-6. SYS Pin Control Register 3 (SYSPC3) (offset = 08h)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKDIN
R-0		R-U

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; -U = Undefined

Table 2-18. SYS Pin Control Register 3 (SYSPC3) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKDIN		ECLK data in. This bit displays the logic state of the ECLK pin when it is configured to be in GIO mode.
		0	The ECLK pin is at logic low (0).
		1	The ECLK pin is at logic high (1).

2.5.1.4 SYS Pin Control Register 4 (SYSPC4)

The SYSPC4 register, shown in Figure 2-7 and described in Table 2-19, controls the logic level output function of the ECLK pin when when it is configured as an output in GIO mode.

Figure 2-7. SYS Pin Control Register 4 (SYSPC4) (offset = 0Ch)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKDOUT
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-19. SYS Pin Control Register 4 (SYSPC4) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKDOUT		ECLK data out write. This bit is only active when the ECLK pin is configured to be in GIO mode. Writes to this bit will only take effect when the ECLK pin is configured as an output in GIO mode. The current logic state of the ECLK pin will be displayed by this bit in both input and output GIO mode.
		0	The ECLK pin is driven to logic low (0).
		1	The ECLK pin is driven to logic high (1).
			Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register.

2.5.1.5 SYS Pin Control Register 5 (SYSPC5)

The SYSPC5 register, shown in Figure 2-8 and described in Table 2-20, controls the set function of the ECLK pin when when it is configured as an output in GIO mode.

Figure 2-8. SYS Pin Control Register 5 (SYSPC5) (offset = 10h)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKSET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-20. SYS Pin Control Register 5 (SYSPC5) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKSET		ECLK data out set. This bit drives the output of the ECLK pin high when set in GIO output mode.
		0	Write: Writing a 0 has no effect.
		1	Write: The ECLK pin is driven to logic high (1).
			Note: The current logic state of the ECPCLKDOUT bit will also be displayed by this bit when the pin is configured in GIO output mode.
			Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register.

2.5.1.6 SYS Pin Control Register 6 (SYSPC6)

The SYSPC6 register, shown in Figure 2-9 and described in Table 2-21, controls the clear function of the ECLK pin when when it is configured as an output in GIO mode..

Figure 2-9. SYS Pin Control Register 6 (SYSPC6) (offset = 14h)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKCLR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-21. SYS Pin Control Register 6 (SYSPC6) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKCLR		ECLK data out clear. This bit drives the output of the ECLK pin low when set in GIO output mode.
		0	Write: The ECLK pin value is unchanged.
		1	Write: The ECLK pin is driven to logic low (0).
			Note: The current logic state of the ECPCLKDOUT bit will also be displayed by this bit when the pin is configured in GIO output mode.
			Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register.

2.5.1.7 SYS Pin Control Register 7 (SYSPC7)

The SYSPC7 register, shown in Figure 2-10 and described in Table 2-22, controls the open drain function of the ECLK pin.

Figure 2-10. SYS Pin Control Register 7 (SYSPC7) (offset = 18h)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKODE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-22. SYS Pin Control Register 7 (SYSPC7) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKODE		ECLK open drain enable. This bit is only active when ECLK is configured to be in GIO mode.
		0	The ECLK pin is configured in push/pull (normal GIO) mode.
		1	The ECLK pin is configured in open drain mode. The ECPCLKDOUT bit in the SYSPC4 register controls the state of the ECLK output buffer:
			ECPCLKDOUT = 0 The ECLK output buffer is driven low
			ECPCLKDOUT = 1 The ECLK output buffer is tristated
			Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register.

2.5.1.8 SYS Pin Control Register 8 (SYSPC8)

The SYSPC8 register, shown in Figure 2-11 and described in Table 2-23, controls the pull enable function of the ECLK pin when it is configured as an input in GIO mode.

Figure 2-11. SYS Pin Control Register 8 (SYSPC8) (offset = 1Ch)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKPUE
R-0		R/W-D

LEGEND: R/W = Read/Write; R = Read only; D = Device Specific; -*n* = value after reset

Table 2-23. SYS Pin Control Register 8 (SYSPC8) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKPUE		ECLK pull enable. Writes to this bit will only take effect when the ECLK pin is configured as an input in GIO mode.
		0	ECLK pull enable is active.
		1	ECLK pull enable is inactive.
			Note: The pull direction (up/down) is selected by the ECPCLKPS bit in the SYSPC9 register.
			Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in input mode by clearing the ECPCLKDIR bit to 0 in the SYSPC2 register.

2.5.1.9 SYS Pin Control Register 9 (SYSPC9)

The SYSPC9 register, shown in Figure 2-12 and described in Table 2-24, controls the pull up/pull down configuration of the ECLK pin when it is configured as an input in GIO mode.

Figure 2-12. SYS Pin Control Register 9 (SYSPC9) (offset = 20h)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKPS
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-24. SYS Pin Control Register 9 (SYSPC9) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKPS		ECLK pull up/pull down select. This bit is only active when ECLK is configured as an input in GIO mode and the pull up/pull down logic is enabled.
		0	ECLK pull down is selected, when pull up/pull down logic is enabled.
		1	ECLK pull up is selected, when pull up/pull down logic is enabled.
			Note: The ECLK pin pull up/pull down logic is enabled by clearing the ECPCLKPUE bit to 0 in the SYSPC8 register.
			Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in input mode by clearing the ECPCLKDIR bit to 0 in the SYSPC2 register.

2.5.1.10 Clock Source Disable Register (CSDIS)

The CSDIS register, shown in Figure 2-13 and described in Table 2-25, controls and displays the state of the device clock sources.

NOTE: Nonimplemented clock sources should not be enabled or used.

Figure 2-13. Clock Source Disable Register (CSDIS) (offset = 30h)

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		CLKSR5OFF	CLKSR4OFF	CLKSR3OFF	Reserved	CLKSR1OFF	CLKSR0OFF
R-0		R/WP-0	R/WP-0	R/WP-1	R-1	R/WP-1	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-25. Clock Source Disable Register (CSDIS) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-3	CLKSR[5-3]OFF		Clock source[5-3] off.
		0	Clock source[5-3] is enabled.
		1	Clock source[5-3] is disabled.
			Note: On wakeup, only clock sources 0, 4, and 5 are enabled.
2	Reserved	1	Reads return 1. Writes have no effect.
1-0	CLKSR[1-0]OFF		Clock source[1-0] off.
		0	Clock source[1-0] is enabled.
		1	Clock source[1-0] is disabled.
			Note: On wakeup, only clock sources 0, 4, and 5 are enabled.

Table 2-26. Clock Sources Table

Clock Source #	Clock Source Name
Clock Source 0	Oscillator
Clock Source 1	PLL
Clock Source 2	Not Implemented
Clock Source 3	EXTCLKIN
Clock Source 4	Low Frequency LPO (Low Power Oscillator) clock
Clock Source 5	High Frequency LPO (Low Power Oscillator) clock
Clock Source 6	Not Implemented
Clock Source 7	Not Implemented

2.5.1.11 Clock Source Disable Set Register (CSDISSET)

The CSDISSET register, shown in Figure 2-14 and described in Table 2-27, sets clock sources to the disabled state.

NOTE: A list of the available clock sources is shown in Table 2-26.

Figure 2-14. Clock Source Disable Set Register (CSDISSET) (offset = 34h)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved		SETCLKSR5 OFF	SETCLKSR4 OFF	SETCLKSR3 OFF	Reserved	SETCLKSR1 OFF	SETCLKSR0 OFF
R-0		R/WP-0	R/WP-0	R/WP-1	R-1	R/WP-1	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-27. Clock Source Disable Set Register (CSDISSET) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-3	SETCLKSR[5-3] OFF	0	Set clock source[5-3] to the disabled state. Read: Clock source[5-3] is enabled. Write: Clock source[5-3] is unchanged.
		1	Read: Clock source[5-3] is disabled. Write: Clock source[5-3] is set to the disabled state. **Note: After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS, CSDISSET, and CSDISCLR registers.**
2	Reserved	1	Reads return 1. Writes have no effect.
1-0	SETCLKSR[1-0] OFF	0	Set clock source[1-0] to the disabled state. Read: Clock source[1-0] is enabled. Write: Clock source[1-0] is unchanged.
		1	Read: Clock source[1-0] is disabled. Write: Clock source[1-0] is set to the disabled state. **Note: After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS, CSDISSET, and CSDISCLR registers.**

2.5.1.12 Clock Source Disable Clear Register (CSDISCLR)

The CSDISCLR register, shown in Figure 2-15 and described in Table 2-28, clears clock sources to the enabled state.

NOTE: A list of the available clock sources is shown in Table 2-26.

Figure 2-15. Clock Source Disable Clear Register (CSDISCLR) (offset = 38h)

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		CLRCLKSR5 OFF	CLRCLKSR4 OFF	CLRCLKSR3 OFF	Reserved	CLRCLKSR1 OFF	CLRCLKSR0 OFF
R-0		R/WP-0	R/WP-0	R/WP-1	R-1	R/WP-1	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-28. Clock Source Disable Clear Register (CSDISCLR) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-3	CLRCLKSR[5-3] OFF		Enables clock source[5-3].
		0	Read: Clock source[5-3] is enabled. Write: Clock source[5-3] is unchanged.
		1	Read: Clock source[5-3] is enabled. Write: Clock source[5-3] is set to the enabled state. **Note: After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS, CSDISSET, and CSDISCLR registers.**
2	Reserved	1	Reads return 1. Writes have no effect.
1-0	CLRCLKSR[1-0] OFF		Enables clock source[1-0].
		0	Read: Clock source[1-0] is enabled. Write: Clock source[1-0] is unchanged.
		1	Read: Clock source[1-0] is enabled. Write: Clock source[1-0] is set to the enabled state. **Note: After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS, CSDISSET, and CSDISCLR registers.**

2.5.1.13 Clock Domain Disable Register (CDDIS)

The CDDIS register, shown in Figure 2-16 and described in Table 2-29, controls the state of the clock domains.

NOTE: All the clock domains are enabled on wakeup.

The application should assure that when HCLK and VCLK_sys are turned off through the HCLKOFF bit, the GCLK1 domain is also turned off.

The register bits in CDDIS are designated as high-integrity bits and have been implemented with error-correcting logic such that each bit, although read and written as a single bit, is actually a multi-bit key with error correction capability. As such, single-bit flips within the “key” can be corrected allowing protection of the system as a whole. An error detected is signaled to the ESM module.

Figure 2-16. Clock Domain Disable Register (CDDIS) (offset = 3Ch)

31							16
Reserved							
R-0							
15			12	11	10	9	8
Reserved				Reserved	Reserved	VCLK_EQEP_OFF	Reserved
R-0				R/WP-0 or 1	R/WP-0 or 1	R/WP-0	R/WP-0
7	6	5	4	3	2	1	0
Reserved	RTICLK1OFF	Reserved	VCLKA1OFF	VCLK2OFF	VCLKPOFF	HCLKOFF	GCLKOFF
R/WP-0 or 1	R/WP-0	R/WP-0 or 1	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-29. Clock Domain Disable Register (CDDIS) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-10	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
9	VCLK_EQEP_OFF		VCLK_EQEP_OFF domain off.
		0	VCLK_EQEP_OFF domain is enabled.
		1	VCLK_EQEP_OFF domain is disabled.
8-7	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
6	RTICLK1OFF		RTICLK1 domain off.
		0	RTICLK1 domain is enabled.
		1	RTICLK1 domain is disabled.
5	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
4	VCLKA1OFF		VCLKA1 domain off.
		0	VCLKA1 domain is enabled.
		1	VCLKA1 domain is disabled.
3	VCLK2OFF		VCLK2 domain off.
		0	VCLK2 domain is enabled.
		1	VCLK2 domain is disabled.
2	VCLKPOFF		VCLK_periph domain off.
		0	VCLK_periph domain is enabled.
		1	VCLK_periph domain is disabled.

Table 2-29. Clock Domain Disable Register (CDDIS) Field Descriptions (continued)

Bit	Field	Value	Description
1	HCLKOFF		HCLK and VCLK_sys domains off.
		0	HCLK and VCLK_sys domains are enabled.
		1	HCLK and VCLK_sys domains are disabled.
0	GCLKOFF		GCLK domain off.
		0	GCLK domain is enabled.
		1	GCLK domain is disabled.

2.5.1.14 Clock Domain Disable Set Register (CDDISSET)

This CDDISSET register, shown in Figure 2-17 and described in Table 2-30, sets clock domains to the disabled state.

Figure 2-17. Clock Domain Disable Set Register (CDDISSET) (offset = 40h)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved				Reserved	Reserved	SETVCLK_EQEP_OFF	Reserved
R-0				R/WP-0 or 1	R/WP-0 or 1	R/WP-0	R/WP-0

7	6	5	4	3	2	1	0
Reserved	SETRTI1CLKOFF	Reserved	SETVCLKA1OFF	SETVCLK2OFF	SETVCLKPOFF	SETHCLKOFF	SETGCLKOFF
R/WP-0 or 1	R/WP-0	R/WP-0 or 1	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-30. Clock Domain Disable Set Register (CDDISSET) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-10	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
9	SETVCLK_EQEP_OFF	0	Set VCLK_EQEP_OFF domain off. Read: VCLK_EQEP_OFF domain is enabled. Write: VCLK_EQEP_OFF domain is unchanged.
		1	Read: VCLK_EQEP_OFF domain is disabled. Write: VCLK_EQEP_OFF domain is set to the enabled state.
8-7	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
6	SETRTI1CLKOFF	0	Set RTICLK1 domain. Read: RTICLK1 domain is enabled. Write: RTICLK1 domain is unchanged.
		1	Read: RTICLK1 domain is disabled. Write: RTICLK1 domain is set to the enabled state.
5	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
4	SETVCLKA1OFF	0	Set VCLKA1 domain. Read: VCLKA1 domain is enabled. Write: VCLKA1 domain is unchanged.
		1	Read: VCLKA1 domain is disabled. Write: VCLKA1 domain is set to the enabled state.
3	SETVCLK2OFF	0	Set VCLK2 domain. Read: VCLK2 domain is enabled. Write: VCLK2 domain is unchanged.
		1	Read: VCLK2 domain is disabled. Write: VCLK2 domain is set to the enabled state.
2	SETVCLKPOFF	0	Set VCLK_periph domain. Read: VCLK_periph domain is enabled. Write: VCLK_periph domain is unchanged.
		1	Read: VCLK_periph domain is disabled. Write: VCLK_periph domain is set to the enabled state.

Table 2-30. Clock Domain Disable Set Register (CDDISSET) Field Descriptions (continued)

Bit	Field	Value	Description
1	SETHCLKOFF		Set HCLK and VCLK_sys domains.
		0	Read: HCLK and VCLK_sys domain is enabled.
			Write: HCLK and VCLK_sys domain is unchanged.
		1	Read: HCLK and VCLK_sys domain is disabled.
			Write: HCLK and VCLK_sys domain is set to the enabled state.
0	SETGCLKOFF		Set GCLK domain.
		0	Read: GCLK domain is enabled.
			Write: GCLK domain is unchanged.
		1	Read: GCLK domain is disabled.
			Write: GCLK domain is set to the enabled state.

2.5.1.15 Clock Domain Disable Clear Register (CDDISCLR)

The CDDISCLR register, shown in Figure 2-18 and described in Table 2-31, clears clock domains to the enabled state.

Figure 2-18. Clock Domain Disable Clear Register (CDDISCLR) (offset = 44h)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		Reserved	Reserved	CLRVCLK_EQEP_OFF	Reserved
R-0		R/WP-0 or 1	R/WP-0 or 1	R/WP-0	R/WP-0

7	6	5	4	3	2	1	0
Reserved	CLRRTI1CLKOFF	Reserved	CLRVCLKA1OFF	CLRVCLK2OFF	CLRVCLKPOFF	CLRHCLKOFF	CLRGCLKOFF
R/WP-0 or 1	R/WP-0	R/WP-0 or 1	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-31. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-10	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
9	CLRVCLK_EQEP_OFF		Clear VCLK_EQEP_OFF domain off.
		0	Read: VCLK_EQEP_OFF domain is enabled.
			Write: VCLK_EQEP_OFF domain is unchanged.
		1	Read: VCLK_EQEP_OFF domain is disabled.
			Write: VCLK_EQEP_OFF domain is set to the enabled state.
8-7	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
6	CLRRTI1CLKOFF		Clear RTICLK1 domain.
		0	Read: RTICLK1 domain is enabled.
			Write: RTICLK1 domain is unchanged.
		1	Read: RTICLK1 domain is disabled.
			Write: RTICLK1 domain is cleared to the enabled state.
5	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.

Table 2-31. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions (continued)

Bit	Field	Value	Description
4	CLRVCLKA1OFF		Clear VCLKA1 domain.
		0	Read: VCLKA1 domain is enabled.
			Write: VCLKA1 domain is unchanged.
		1	Read: VCLKA1 domain is disabled.
			Write: VCLKA1 domain is cleared to the enabled state.
3	CLRVCLK2OFF		Clear VCLK2 domain.
		0	Read: VCLK2 domain is enabled.
			Write: VCLK2 domain is unchanged.
		1	Read: VCLK2 domain is disabled.
			Write: VCLK2 domain is cleared to the enabled state.
2	CLRVCLKPOFF		Clear VCLK_periph domain.
		0	Read: VCLK_periph domain is enabled.
			Write: VCLK_periph domain is unchanged.
		1	Read: VCLK_periph domain is disabled.
			Write: VCLK_periph domain is cleared to the enabled state.
1	CLRHCLKOFF		Clear HCLK and VCLK_sys domains.
		0	Read: HCLK and VCLK_sys domain is enabled.
			Write: HCLK and VCLK_sys domain is unchanged.
		1	Read: HCLK and VCLK_sys domain is disabled.
			Write: HCLK and VCLK_sys domain is cleared to the enabled state.
0	CLRGCLKOFF		Clear GCLK domain.
		0	Read: GCLK domain is enabled.
			Write: GCLK domain is unchanged.
		1	Read: GCLK domain is disabled.
			Write: GCLK domain is cleared to the enabled state.

2.5.1.16 GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC)

The GHVSRC register, shown in Figure 2-19 and described in Table 2-32, controls the clock source configuration for the GCLK, HCLK, VCLK and VCLK2 clock domains.

NOTE: Nonimplemented clock sources should not be enabled or used. A list of the available clock sources is shown in Table 2-26.

Figure 2-19. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) (offset = 48h)

31	28	27	24	23	20	19	16
Reserved		GHVWAKE		Reserved		HVLPM	
R-0		R/WP-0		R-0		R/WP-0	

15	4	3	0
Reserved		GHVSRC	
R-0		R/WP-0	

LEGEND: R = Read only; R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 2-32. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions

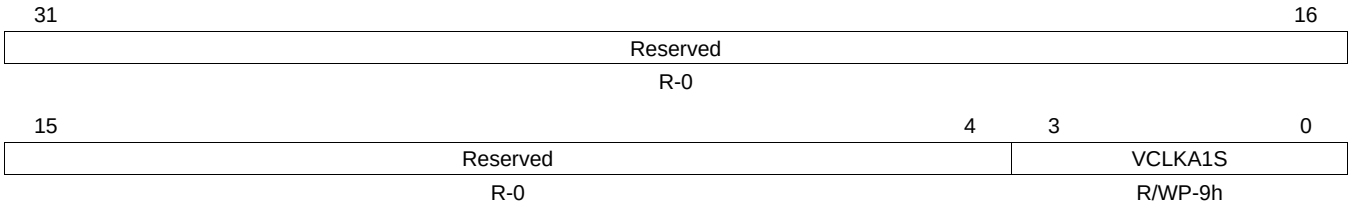
Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	GHVWAKE		GCLK, HCLK, VCLK, VCLK2 source on wakeup.
		0	Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		1h	Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		2h	Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		3h	Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		4h	Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		5h	Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		6h	Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		7h	Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		8h-Fh	Reserved
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	HVLPM		HCLK, VCLK, VCLK2 source on wakeup when GCLK is turned off.
		0	Clock source0 is the source for HCLK, VCLK, VCLK2 on wakeup.
		1h	Clock source1 is the source for HCLK, VCLK, VCLK2 on wakeup.
		2h	Clock source2 is the source for HCLK, VCLK, VCLK2 on wakeup.
		3h	Clock source3 is the source for HCLK, VCLK, VCLK2 on wakeup.
		4h	Clock source4 is the source for HCLK, VCLK, VCLK2 on wakeup.
		5h	Clock source5 is the source for HCLK, VCLK, VCLK2 on wakeup.
		6h	Clock source6 is the source for HCLK, VCLK, VCLK2 on wakeup.
		7h	Clock source7 is the source for HCLK, VCLK, VCLK2 on wakeup.
		8h-Fh	Reserved
15-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	GHVSRC		GCLK, HCLK, VCLK, VCLK2 current source. **Note: The GHVSRC[3-0] bits are updated with the HVLPM[3-0] setting when GCLK is turned off, and are updated with the GHVWAKE[3-0] setting on system wakeup.**
		0	Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2.
		1h	Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2.
		2h	Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2.
		3h	Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2.
		4h	Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2.
		5h	Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2.
		6h	Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2.
		7h	Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2.
		8h-Fh	Reserved

2.5.1.17 Peripheral Asynchronous Clock Source Register (VCLKASRC)

The VCLKASRC register, shown in Figure 2-20 and described in Table 2-33, sets the clock source for the asynchronous peripheral clock domains to be configured to run from a specific clock source.

NOTE: Nonimplemented clock sources should not be enabled or used. A list of the available clock sources is shown in Table 2-26.

Figure 2-20. Peripheral Asynchronous Clock Source Register (VCLKASRC) (offset = 4Ch)



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-33. Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	VCLKA1S		Peripheral asynchronous clock1 source.
		0	Clock source0 is the source for peripheral asynchronous clock1.
		1h	Clock source1 is the source for peripheral asynchronous clock1.
		2h	Clock source2 is the source for peripheral asynchronous clock1.
		3h	Clock source3 is the source for peripheral asynchronous clock1.
		4h	Clock source4 is the source for peripheral asynchronous clock1.
		5h	Clock source5 is the source for peripheral asynchronous clock1.
		6h	Clock source6 is the source for peripheral asynchronous clock1.
		7h	Clock source7 is the source for peripheral asynchronous clock1.
		8h-Fh	VCLK is the source for peripheral asynchronous clock1.

2.5.1.18 RTI Clock Source Register (RCLKSRC)

The RCLKSRC register, shown in Figure 2-21 and described in Table 2-34, controls the RTI (Real Time Interrupt) clock source selection.

NOTE: Important constraint when the RTI clock source is not VCLK

If the RTIx clock source is chosen to be anything other than the default VCLK, then the RTI clock needs to be at least three times slower than the VCLK. This can be achieved by configuring the RTIxCLK divider in this register. This divider is internally bypassed when the RTIx clock source is VCLK.

NOTE: A list of the available clock sources is shown in Table 2-26.

Figure 2-21. RTI Clock Source Register (RCLKSRC) (offset = 50h)

31	26	25	24	23	20	19	16
Reserved		Reserved		Reserved		Reserved	
R-0		R/WP-1h		R-0		R/WP-9h	

15	10	9	8	7	4	3	0
Reserved		RTI1DIV		Reserved		RTI1SRC	
R-0		R/WP-1h		R-0		R/WP-9h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-34. RTI Clock Source Register (RCLKSRC) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reads return 0. Writes have no effect.
25-24	Reserved	1h	Reads return value and privilege mode writes allowed.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	Reserved	9h	Reads return value and privilege mode writes allowed.
15-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	RTI1DIV		RTI clock1 Divider.
		0	RTICLK1 divider value is 1.
		1h	RTICLK1 divider value is 2.
		2h	RTICLK1 divider value is 4.
		3h	RTICLK1 divider value is 8.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	RTI1SRC		RTI clock1 source.
		0	Clock source0 is the source for RTICLK1.
		1h	Clock source1 is the source for RTICLK1.
		2h	Clock source2 is the source for RTICLK1.
		3h	Clock source3 is the source for RTICLK1.
		4h	Clock source4 is the source for RTICLK1.
		5h	Clock source5 is the source for RTICLK1.
		6h	Clock source6 is the source for RTICLK1.
		7h	Clock source7 is the source for RTICLK1.
		8h-Fh	VCLK is the source for RTICLK1.

2.5.1.19 Clock Source Valid Status Register (CSVSTAT)

The CSVSTAT register, shown in Figure 2-22 and described in Table 2-35, indicates the status of usable clock sources.

NOTE: A list of the available clock sources is shown in Table 2-26.

Figure 2-22. Clock Source Valid Status Register (CSVSTAT) (offset = 54h)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved		CLKSR5V	CLKSR4V	CLKSR3V	Reserved	CLKSR1V	CLKSR0V
R-0		R-0	R-1	R-1	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 2-35. Clock Source Valid Register (CSVSTAT) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-3	CLKSR[5-3]V		Clock source[5-3] valid.
		0	Clock source[5-3] is not valid.
		1	Clock source[5-3] is valid.
			Note: If the valid bit of the source of a clock domain is not set (that is, the clock source is not fully stable), the respective clock domain is disabled by the Global Clock Module (GCM).
2	Reserved	0	Reads return 0. Writes have no effect.
1-0	CLKSR[1-0]V		Clock source[1–0] valid.
		0	Clock source[1–0] is not valid.
		1	Clock source[1–0] is valid.
			Note: If the valid bit of the source of a clock domain is not set (that is, the clock source is not fully stable), the respective clock domain is disabled.

2.5.1.20 Memory Self-Test Global Control Register (MSTGCR)

The MSTGCR register, shown in Figure 2-23 and described in Table 2-36, controls several aspects of the PBIST (Programmable Built-In Self Test) memory controller.

Figure 2-23. Memory Self-Test Global Control Register (MSTGCR) (offset = 58h)

31	16
Reserved	
R-0	

15	10	9	8	7	4	3	0
Reserved		ROM_DIV		Reserved		MSTGENA	
R-0		R/WP-0		R-0		R/WP-5h	

LEGEND: R = Read only; R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 2-36. Memory Self-Test Global Control Register (MSTGCR) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	ROM_DIV		Prescaler divider bits for ROM clock source.
		0	ROM clock source is HCLK divided by 1. PBIST will reset for 16 VBUS cycles.
		1h	ROM clock source is HCLK divided by 2. PBIST will reset for 32 VBUS cycles.
		2h	ROM clock source is HCLK divided by 4. PBIST will reset for 64 VBUS cycles.
		3h	ROM clock source is HCLK divided by 8. PBIST will reset for 96 VBUS cycles.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MSTGENA		Memory self-test controller global enable key **Note: Enabling the MSTGENA key will generate a reset to the state machine of the PBIST controller.**
		Ah	Memory self-test controller is enabled.
		All other values	Memory self-test controller is disabled. **Note: It is recommended that a value of 5h be used to disable the memory self-test controller. This value will give maximum protection from a bit flip inducing event that would inadvertently enable the controller.**

2.5.1.21 Memory Hardware Initialization Global Control Register (MINITGCR)

The MINITGCR register, shown in Figure 2-24 and described in Table 2-37, enables automatic hardware memory initialization.

Figure 2-24. Memory Hardware Initialization Global Control Register (MINITGCR) (offset = 5Ch)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		MINITGENA	
R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-37. Memory Hardware Initialization Global Control Register (MINITGCR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MINITGENA		Memory hardware initialization global enable key.
		Ah	Global memory hardware initialization is enabled.
		All other values	Global memory hardware initialization is disabled.
			Note: It is recommended that a value of 5h be used to disable memory hardware initialization. This value will give maximum protection from an event that would inadvertently enable the controller.

2.5.1.22 PBIST Controller/ Memory Initialization Enable Register (MSINENA)

The MSINENA register, shown in Figure 2-25 and described in Table 2-38, enables PBIST controllers for memory self test and the memory modules initialized during automatic hardware memory initialization.

Figure 2-25. PBIST Controller/Memory Initialization Enable Register (MSINENA) (offset = 60h)

31	0
MSIENA[31:0]	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-38. PBIST Controller/Memory Initialization Enable Register (MSINENA) Field Descriptions

Bit	Field	Value	Description
31-0	MSIENA		PBIST controller and memory initialization enable register. In memory self-test mode, all the corresponding bits of the memories to be tested should be set before enabling the global memory self-test controller key (MSTGENA) in the MSTGCR register (offset 58h). The reason for this is that MSTGENA, in addition to being the global enable for all individual PBIST controllers, is the source for the reset generation to all the PBIST controller state machines. Disabling the MSTGENA or MINITGENA key (by writing from Ah to any other value) will reset all the MSIENA [31-0] bits to their default values. **Note: Software should ensure that both the memory self-test global enable key (MSTGENA) and the memory hardware initialization global key (MINITGENA) are not enabled at the same time.**
		0	**In memory self-test mode (MSTGENA = Ah):** PBIST controller [31-0] is disabled. **In memory Initialization mode (MINITGENA = Ah):** Memory module [31-0] auto hardware initialization is disabled.
		1	**In memory self-test mode (MSTGENA = Ah):** PBIST controller [31-0] is enabled. **In memory Initialization mode (MINITGENA = Ah):** Memory module [31-0] auto hardware initialization is enabled.

2.5.1.23 Memory Self-Test Fail Status Register (MSTFAIL)

The MSTFAIL register, shown in Figure 2-26 and described in Table 2-39, shows the fail status of the memory self-tests.

Figure 2-26. Memory Self-Test Fail Status Register (MSTFAIL) (offset = 64h)

31	0
MSTF[31:0]	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-39. Memory Self-Test Fail Status Register (MSTFAIL) Field Descriptions

Bit	Field	Value	Description
31-0	MSTF		Memory self-test fail status bit. **Note: Disabling the MSTGENA key (by writing from Ah to any other value) will reset all the individual fail status bits to their default values.**
		0	Read: PBIST controller [31-0] run did not fail. Write: A write of 0 has no effect.
		1	Read: PBIST controller [31-0] run failed. Write: The bit is cleared to 0.

2.5.1.24 MSTC Global Status Register (MSTCGSTAT)

The MSTCGSTAT register, shown in Figure 2-27 and described in Table 2-40, shows the status of the memory hardware initialization and the memory seft-test.

Figure 2-27. MSTC Global Status Register (MSTCGSTAT) (offset = 68h)

31	16
Reserved	
R-0	

15	9	8	7	1	0
Reserved		MINIDONE	Reserved		MSTDONE
R-0		R/W1CP-0	R-0		R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; WP = Write in privileged mode only; -*n* = value after reset

Table 2-40. MSTC Global Status Register (MSTCGSTAT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	MINIDONE		Memory hardware initialization complete status. **Note: Disabling the MINITGENA key (by writing from Ah to any other value) will clear the MINIDONE status bit to 0.** **Note: Individual memory initialization status is shown in the MINISTAT register.**
		0	Read: Memory hardware initialization is not complete for all memory. Write: A write of 0 has no effect.
		1	Read: Hardware initialization of all memory is completed. Write: The bit is cleared to 0.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	MSTDONE		Memory self-test run complete status. **Note: Disabling the MSTGENA key (by writing from Ah to any other value) will clear the MSTDONE status bit to 0.**
		0	Read: Memory self-test is not completed. Write: A write of 0 has no effect.
		1	Read: Memory self-test is completed. Write: The bit is cleared to 0.

2.5.1.25 Memory Hardware Initialization Status Register (MINISTAT)

The MINISTAT register, shown in Figure 2-28 and described in Table 2-41, indicates the status of hardware memory initialization.

Figure 2-28. Memory Hardware Initialization Status Register (MINISTAT) (offset = 6Ch)

31	0
MIDONE[31:0]	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-41. Memory Hardware Initialization Status Register (MINISTAT) Field Descriptions

Bit	Field	Value	Description
31-0	MIDONE		Memory hardware initialization status bit.
		0	Read: Memory module[31-0] hardware initialization is not completed. Write: A write of 0 has no effect.
		1	Read: Memory module[31-0] hardware initialization is completed. Write: The bit is cleared to 0. **Note: Disabling the MINITGENA key (by writing from Ah to any other value) will reset all the individual status bits to 0.**

2.5.1.26 PLL Control Register 1 (PLLCTL1)

The PLLCTL1 register, shown in Figure 2-29 and described in Table 2-42, controls the output frequency of PLL (Clock Source 1 - FMzPLL). It also controls the behavior of the device if a PLL slip or oscillator failure is detected.

Figure 2-29. PLL Control Register 1 (PLLCTL1) (offset = 70h)

31	30 29	28 24	23	22	21 16
ROS	MASK_SLIP	PLLDIV	ROF	Rsvd	REFCLKDIV
R/WP-0	R/WP-1h	R/WP-Fh	R/WP-0	R-0	R/WP-2h

15 0
PLLMUL
R/WP-5F00h

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-42. PLL Control Register 1 (PLLCTL1) Field Descriptions

Bit	Field	Value	Description
31	ROS		Reset on PLL Slip
		0	Do not reset system when PLL slip is detected
		1	Reset when PLL slip is detected
			Note: BPOS (Bits 30-29) must also be enabled for ROS to be enabled.
30-29	MASK_SLIP		Mask detection of PLL slip
		2h	Bypass on PLL Slip is disabled. If a PLL Slip is detected no action is taken.
		All other values	Bypass on PLL Slip is enabled. If a PLL Slip is detected the device will automatically bypass the PLL and use the oscillator to provide the device clock.
			Note: If ROS (Bit 31) is set to 1 the device will be reset if a PLL Slip and the PLL will be bypassed after the reset occurs.
28-24	PLLDIV		PLL Output Clock Divider
			R = PLLDIV + 1 $f_{PLL\ CLK} = f_{post\_ODCLK} / R$
		0	$f_{PLL\ CLK} = f_{post\_ODCLK} / 1$
		1h	$f_{PLL\ CLK} = f_{post\_ODCLK} / 2$
		:	:
		1Fh	$f_{PLL\ CLK} = f_{post\_ODCLK} / 32$
23	ROF		Reset on Oscillator Fail
		0	Do not reset system when oscillator is out of range.
		1	Reset when oscillator is out of range.
22	Reserved	0	Value has no effect on PLL operation.
21-16	REFCLKDIV		Reference Clock Divider
			NR = REFCLKDIV + 1
			$f_{INT\ CLK} = f_{OSCIN} / NR$
		0	$f_{INT\ CLK} = f_{OSCIN} / 1$
		1h	$f_{INT\ CLK} = f_{OSCIN} / 2$
		:	:
		3Fh	$f_{INT\ CLK} = f_{OSCIN} / 64$

Table 2-42. PLL Control Register 1 (PLLCTL1) Field Descriptions (continued)

Bit	Field	Value	Description
15-0	PLLMUL		PLL Multiplication Factor
			Valid multiplication factors are from 1 to 256.
			NF = (PLLMUL / 256) + 1
			$f_{VCO\ CLK} = f_{INT\ CLK} \times NF$
		0h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 1$
		100h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 2$
		:	:
		5B00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 92$
		5C00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 93$
		:	:
		FF00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 256$

2.5.1.27 PLL Control Register 2 (PLLCTL2)

The PLLCTL2 register, shown in Figure 2-30 and described in Table 2-43, controls the modulation characteristics and the output divider of the PLL.

Figure 2-30. PLL Control Register 2 (PLLCTL2) (offset = 74h)

31	30	22	21	20	16
FMENA	SPREADINGRATE		Rsvd	MULMOD	
R/WP-0	R/WP-1FFh		R-0	R/WP-0	

15	12	11	9	8	0
MULMOD		ODPLL		SPR_AMOUNT	
R/WP-7h		R/WP-1h		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-43. PLL Control Register 2 (PLLCTL2) Field Descriptions

Bit	Field	Value	Description
31	FMENA		Frequency Modulation Enable.
		0	Disable frequency modulation
		1	Enable frequency modulation
30-22	SPREADINGRATE		NS = SPREADINGRATE + 1
			$f_{mod} = f_s = f_{INT\ CLK}/(2 \times NS)$
		0	$f_{mod} = f_s = f_{INT\ CLK} / (2 \times 1)$
		1h	$f_{mod} = f_s = f_{INT\ CLK} / (2 \times 2)$
		:	:
		1FFh	$f_{mod} = f_s = f_{INT\ CLK} / (2 \times 512)$
21	Reserved	0	Value has no effect on PLL operation.
20-12	MULMOD		Multiplier Correction when Frequency Modulation is enabled
			When FMENA = 0, MUL_when_MOD = 0; When FMENA = 1, MUL_when_MOD = (MULMOD / 256)
		0	No adder to NF
		8h	MUL_when_MOD = 8/256
		9h	MUL_when_MOD = 9/256
		:	:
		1FFh	MUL_when_MOD = 511/256
11-9	ODPLL		Internal PLL Output Divider.
			OD = ODPLL + 1
			$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/OD$
			Note that the PLL output clock is gated off if ODPLL is changed while the PLL is active.
		0	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 1$
		1h	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 2$
		:	:
		7h	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 8$
8-0	SPR_AMOUNT		Spreading Amount.
			NV = (SPR_AMOUNT + 1)/2048
			NV ranges from 1/2048 to 512/2048
			Note that the PLL output clock is disabled for 1 modulation period if the SPR_AMOUNT field is changed while the frequency modulation is enabled. If frequency modulation is disabled and SPR_AMOUNT is changed, there is no effect on the PLL output clock.
		0	NV = 1/2048
		1h	NV = 2/2048
		:	:
		1FFh	NV = 512/2048

2.5.1.28 SYS Pin Control Register 10 (SYSPC10)

The SYSPC10 register, shown in Figure 2-31 and described in Table 2-44, controls the function of the ECLK slew mode.

Figure 2-31. SYS Pin Control Register 10 (SYSPC10) (offset = 78h)

31	16
Reserved	
R-0	

15	1	0
Reserved		ECLK_SLEW
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-44. SYS Pin Control Register 10 (SYSPC10) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECLK_SLEW		ECLK slew control. This bit controls between the fast or slow slew mode.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.

2.5.1.29 Die Identification Register Lower Word (DIEIDL)

The DIEIDL register, shown in Figure 2-32 and described in Table 2-45, contains information about the die wafer number, and X, Y wafer coordinates.

Figure 2-32. Die Identification Register, Lower Word (DIEIDL) [offset = 7Ch]

31	24	23	16
WAFER #		Y WAFER COORDINATE	
R-D		R-D	

15	12	11	0
Y WAFER COORDINATE		X WAFER COORDINATE	
R-D		R-D	

LEGEND: R = Read only; -*n* = value after reset; -D = device specific

Table 2-45. Die Identification Register, Lower Word (DIEIDL) Field Descriptions

Bit	Field	Description
31-24	WAFER #	These read-only bits contain the wafer number of the device.
23-12	Y WAFER COORDINATE	These read-only bits contain the Y wafer coordinate of the device.
11-0	X WAFER COORDINATE	These read-only bits contain the X wafer coordinate of the device.

NOTE: Die Identification Information

The die identification information will vary from unit to unit. This information is programmed by TI as part of the initial device test procedure.

2.5.1.30 Die Identification Register Upper Word (DIEIDH)

The DIEIDH register, shown in Figure 2-33 and described in Table 2-46, contains information about the die lot number.

Figure 2-33. Die Identification Register, Upper Word (DIEIDH) [offset = 80h]

31	24	23	16
Reserved		LOT #	
R-0		R-D	

15	0
LOT #	
R-D	

LEGEND: R/W = Read/Write; R = Read only; D = Value is device dependent; -*n* = value after reset

Table 2-46. Die Identification Register, Upper Word (DIEIDH) Field Descriptions

Bit	Field	Description
31-24	Reserved	Reserved for TI use. Writes have no effect.
23-0	LOT #	This read-only register contains the device lot number.

NOTE: Die Identification Information

The die identification information will vary from unit to unit. This information is programmed by TI as part of the initial device test procedure.

2.5.1.31 LPO/Clock Monitor Control Register (LPOMONCTL)

The LPOMONCTL register, shown in Figure 2-34 and described in Table 2-47, controls the Low Frequency (Clock Source 4) and High Frequency (Clock Source 5) Low Power Oscillator's trim values.

Figure 2-34. LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 88h]

31	25	24	23	17	16
Reserved		BIAS ENABLE	Reserved		OSCFRQCONFIGCNT
R-0		R/WP-1	R-0		R/WP-0

15	13	12	8	7	5	4	0
Reserved		HFTRIM		Reserved		LFTRIM	
R-0		R/WP-10h		R-0		R/WP-10h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-47. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Read returns 0. Writes have no effect.
24	BIAS ENABLE		Bias enable. Reset by nPORRST.
		0	The bias circuit inside the low-power oscillator (LPO) is disabled.
		1	The bias circuit inside LPO is enabled.
23-17	Reserved	0	Read returns 0. Writes have no effect.
16	OSCFRQCONFIGCNT		Configures the counter based on OSC frequency.
		0	Read: OSC freq is ≤ 20MHz
			Write: A write of 0 has no effect.
		1	Read: OSC freq is > 20MHz and ≤ 80MHz
			Write: A write of 1 has no effect.
15-13	Reserved	0	Read returns 0. Writes have no effect.

Table 2-47. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (continued)

Bit	Field	Value	Description
12-8	HFTRIM		High-frequency oscillator trim value. This five-bit value is used to center the HF oscillator's frequency. Reset by nPORRST. **Caution: This value should only be changed when the HF oscillator is not the source for a clock domain; otherwise, a system failure could result.** **The following HF TRIM values reflect for the F021 process.** The following values are the ratio, f / fo, expressed as a percent:
		0	29.52
		1h	34.24
		2h	38.85
		3h	43.45
		4h	47.99
		5h	52.55
		6h	57.02
		7h	61.46
		8h	65.92
		9h	70.17
		Ah	74.55
		Bh	78.92
		Ch	83.17
		Dh	87.43
		Eh	91.75
		Fh	95.89
		10h	100.00% (Default at Reset)
		11h	104.09
		12h	108.17
		13h	112.32
		14h	116.41
		15h	120.67
		16h	124.42
		17h	128.38
		18h	132.24
		19h	136.15
		1Ah	140.15
		1Bh	143.94
		1Ch	148.02
		1Dh	151.80
		1Eh	155.50
		1Fh	159.35
7-5	Reserved	0	Read returns 0. Writes have no effect.

Table 2-47. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (continued)

Bit	Field	Value	Description
4-0	LFTRIM		Low-frequency oscillator trim value. This five-bit value is used to center the LF oscillator's frequency. Reset by nPORRST. **Caution: This value should only be changed when the LF oscillator is not the source for a clock domain; otherwise, a system failure could result.** **The following LF TRIM values reflect for the F021 process.** The following values are the ratio, f / fo, expressed as a percent:
		0	20.67
		1h	25.76
		2h	30.84
		3h	35.90
		4h	40.93
		5h	45.95
		6h	50.97
		7h	55.91
		8h	60.86
		9h	65.78
		Ah	70.75
		Bh	75.63
		Ch	80.61
		Dh	85.39
		Eh	90.23
		Fh	95.11
		10h	100.00% (Default at Reset)
		11h	104.84
		12h	109.51
		13h	114.31
		14h	119.01
		15h	123.75
		16h	128.62
		17h	133.31
		18h	138.03
		19h	142.75
		1Ah	147.32
		1Bh	152.02
		1Ch	156.63
		1Dh	161.38
		1Eh	165.90
		1Fh	170.42

2.5.1.32 Clock Test Register (CLKTEST)

The CLKTEST register, shown in Figure 2-35 and described in Table 2-48, controls the clock signal that is supplied to the ECLK pin for test and debug purposes.

NOTE: Clock Test Register Usage

This register should only be used for test and debug purposes.

NOTE: Nonimplemented clock sources should not be enabled or used.

Figure 2-35. Clock Test Register (CLKTEST) (offset = 8Ch)

31	27	26	25	24
Reserved		ALTLIMPCLOCK ENABLE	RANGEDET CTRL	RANGEDET ENABLE
R-0		R/WP-0	R/WP-0	R/WP-0

23	20	19	16
Reserved		CLK_TEST_EN	
R-0		R/WP-Ah	

15	12	11	8	7	4	3	0
Reserved		SEL_N2HET_PIN		Reserved		SEL_ECP_PIN	
R-0		R/WP-0		R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-48. Clock Test Register (CLKTEST) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26	ALTLIMPCLOCKENABLE		This bit selects a clock driven by the GIOB[0] pin as an alternate limp clock to the clock monitor phase frequency detect (PFD).
		0	The 10-MHz LPO fast clock is the compare clock for the clock detect PFD circuit and the source to limp clock on a clock fail.
		1	The ALTLIMPCLOCK driven on the GIOB[0] pin is the compare clock for the clock detect PFD circuit and the source to limp clock on a clock fail.
25	RANGEDETCTRL		Range detection control. This bit's functionality is dependant on the state of the RANGEDETENSSEL bit (Bit 24) of the CLKTEST register.
		0	The clock monitor range detection circuitry (RANGEDETECTENABLE) is disabled.
		1	The clock monitor range detection circuitry (RANGEDETECTENABLE) is enabled.
24	RANGEDETENABLE		Selects range detection enable. This bit resets asynchronously on power on reset.
		0	The range detect enable is generated by the hardware in the clock monitor wrapper.
		1	The range detect enable is controlled by the RANGEDETCTRL bit (Bit 25) of the CLKTEST register.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	CLK_TEST_EN		Clock test enable. This bit enables the clock going to the ECLK pin. **Note: The ECLK pin must also be placed into Functional mode by setting the ECPCLKFUN bit to 1 in the SYSPC1 register.**
		5h	Clock going to ECLK pin is enabled.
		All other values	Clock going to ECLK pin is disabled.
15-12	Reserved	0	Reads return 0. Writes have no effect.

Table 2-48. Clock Test Register (CLKTEST) Field Descriptions (continued)

Bit	Field	Value	Description
11-8	SEL_N2HET_PIN		N2HET[2] pin clock source valid, clock source select
		0	Oscillator valid status
		1h	PLL1 valid status
		2h-4h	Reserved
		5h	[CLK10M] High-frequency LPO (Low-Power Oscillator) clock valid status
		6h-7h	Reserved
		8h	[CLK80K] Low-frequency LPO (Low-Power Oscillator) clock valid status
		9h-Fh	Oscillator valid status
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	SEL_ECP_PIN		ECLK pin clock source select **Note: Only valid clock sources can be selected for the ECLK pin. Valid clock sources are displayed by the CSVSTAT register.**
		0	Oscillator clock
		1h	PLL1 clock output
		2h-3h	Reserved
		4h	[CLK80K] Low-frequency LPO (Low-Power Oscillator) clock
		5h	[CLK10M] High-frequency LPO (Low-Power Oscillator) clock
		6h-7h	Reserved
		8h	GCLK
		9h	RTI Base
		Ah	Reserved
		Bh	VCLKA1
		Ch-Eh	Reserved
		Fh	Flash HD Pump Oscillator

2.5.1.33 DFT Control Register (DFTCTRLREG)

This register is shown in Figure 2-36 and described in Table 2-49.

Figure 2-36. DFT Control Register (DFTCTRLREG) (offset = 90h)

31	16
Reserved	
R-0	

15	14	13	12	11	10	9	8	7	4	3	0
Reserved		DFTWRITE		Reserved		DFTREAD		Reserved		TEST_MODE_KEY	
R-0		R/WP-1h		R-0		R/WP-1h		R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-49. DFT Control Register (DFTCTRLREG) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reads return 0. Writes have no effect.
13-12	DFTWRITE	0-3h	DFT logic access. See Table 2-50.
11-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	DFTREAD	0-3h	DFT logic access. See Table 2-50.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	TEST_MODE_KEY		Test mode key. This register is for internal TI use only.
		Ah	Register key enable. ALL the bits can be written to only when the key is enabled. On reset, these bits will be set to 5h.
		All other values	Register key disable. All bits in this register will maintain their default value and cannot be written.

Table 2-50. DFT Logic Access Mode

DFTWRITE		DFTREAD		
Bit 13	Bit 12	Bit 9	Bit 8	Mode
0	0	0	0	Configured in stress mode
0	1	0	1	Configured in slow mode
1	0	1	0	Configured in fast mode
1	1	1	1	Configured in screen mode

2.5.1.34 DFT Control Register 2 (DFTCTRLREG2)

This register is shown in Figure 2-37 and described in Table 2-51.

Figure 2-37. DFT Control Register 2 (DFTCTRLREG2) (offset = 94h)

31	16
IMPDF[27:12]	
R/WP-0	

15	4	3	0
IMPDF[11:0]		TEST_MODE_KEY	
R/WP-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-51. DFT Control Register 2 (DFTCTRLREG2) Field Descriptions

Bit	Field	Value	Description
31-4	IMPDF		DFT Implementation defined bits.
		0	IMPDF is disabled.
		1	IMPDF is enabled.
3-0	TEST_MODE_KEY		Test mode key. This register is for internal TI use only.
		Ah	Register key enable. All the bits can be written to only when the key is enabled.
		All other values	Register key disable. All bits in this register will maintain their default value and cannot be written.

2.5.1.35 General Purpose Register (GPREG1)

This register is shown in Figure 2-38 and described in Table 2-52.

Figure 2-38. General Purpose Register (GPREG1) (offset = A0h)

31	26	25	20	19	16
Reserved		PLL1_FBSLIP_FILTER_COUNT		PLL1_FBSLIP_FILTER_KEY	
R-0		R/WP-0		R/WP-5h	

15	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-52. General Purpose Register (GPREG1) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reads return 0. Writes have no effect.
25-20	PLL1_FBSLIP_FILTER_ COUNT	0-3Fh	FBSLIP down counter programmed value. Configures the system response when a FBSLIP is indicated by the PLL macro. When PLL1_FBSLIP_FILTER_KEY is not Ah, the down counter counts from the programmed value on every LPO high-frequency clock once PLL macro indicates FBSLIP. When the count reaches 0, if the synchronized FBSLIP signal is still high, an FBSLIP condition is indicated to the system module and is captured in the global status register. When the FBSLIP signal from the PLL macro is de-asserted before the count reaches 0, the counter is reloaded with the programmed value. On reset, counter value is 0. Counter must be programmed to a non-zero value and enabled for the filtering to be enabled.
		0	Filtering is disabled.
		1h	Filtering is enabled. Every slip is recognized.
		2h	Filtering is enabled. The slip must be at least 2 HF LPO cycles wide in order to be recognized as a slip.
		:	:
		3Fh	Filtering is enabled. The slip must be at least 63 HF LPO cycles wide in order to be recognized as a slip.
19-16	PLL1_FBSLIP_FILTER_ KEY		Enable the FBSLIP filtering.
		5h	On reset, the FBSLIP filter is disabled and the FBSLIP passes through.
		Fh	This is an unsupported value. You should avoid writing this value to this bit field.
		All other values	FBSLIP filtering is enabled. Recommended to program Ah in this bit field. Enabling of the FBSLIP occurs when the KEY is programmed and a non-zero value is present in the COUNT field.
15-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.1.36 Imprecise Fault Status Register (IMPFASTS)

The IMPFASTS register, shown in Figure 2-39 and described in Table 2-53, displays information about imprecise aborts that have occurred.

Figure 2-39. Imprecise Fault Status Register (IMPFASTS) (offset = A8h)

31	24	23	16
Reserved		MASTERID	
R-0		R-0	

15	10	9	8	7	1	0
Reserved		NCBA	VBUSA	Reserved		ATYPE
R-0		R-0	R-0	R-0		R/WC-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 2-53. Imprecise Fault Status Register (IMPFASTS) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	MASTERID	0-FFh	Master ID. This register indicates which master is responsible for the imprecise abort. The master ID value depends on device implementation- see Table 2-2 for MASTERID values for each bus master. **Notes:** • These bits are only updated when an imprecise abort occurs • These bits are cleared to 0 only on power-on reset. The value of these bits remains unchanged after all other resets.
15-10	Reserved	0	Reads return 0. Writes have no effect.
9	NCBA		Non-cacheable, bufferable abort (NCBA). This register indicates the imprecise abort was generated by a non-cacheable, bufferable write or shared device write through the write buffer of the CPU. **Notes:** • This bit is only updated when an imprecise abort generated by a non-cacheable, bufferable write or shared device write occurs. • This bit is cleared to 0 only on power-on reset. The value of this register remains unchanged after all other resets.
		0	A NCBA is not responsible for the last imprecise abort.
		1	A NCBA was written with an illegal address and generated an imprecise abort.
8	VBUSA		VBUS abort. This register indicates the imprecise abort was generated when writing into the peripheral frame. **Notes:** • This bit is only updated when an imprecise abort is generated when writing into the peripheral frame • This bit is cleared to 0 only on power-on reset. The value of this register remains unchanged after all other resets.
		0	The peripheral frame did not generate the last imprecise abort.
		1	The peripheral frame was written with an illegal address and generated an imprecise abort.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	ATYPE		Abort type.This bit indicates to the CPU whether the last abort was an imprecise abort or a precise abort. **Notes:** • This bit is updated after each abort is generated to the CPU. • This bit is cleared on CPU read. • This bit is cleared to 0 only on power-on reset. The value of this bit remains unchanged after all other resets
		0	The last abort generated was a precise abort. MASTERID, VBUSA, NCBA, and IMPFTADD were not updated.
		1	The last abort generated was an imprecise abort. MASTERID, VBUSA, NCBA, and IMPFTADD were updated. **Note: Once ATYPE is set, the IMPFAWADD and IMPFASTS bits are not updated by subsequent ABORT signals.**

2.5.1.37 Imprecise Fault Address Register (IMPFTADD)

This IMPFTADD register, shown in Figure 2-40 and described in Table 2-54, shows the address at which an imprecise abort occurred.

Figure 2-40. Imprecise Fault Write Address Register (IMPFTADD) (offset = ACh)

31	0
IMPFTADD	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 2-54. Imprecise Fault Write Address Register (IMPFTADD) Field Descriptions

Bit	Field	Description
31-0	IMPFTADD	These bits contain the fault address when an imprecise abort occurs. **Note: These bits are only updated when an imprecise abort occurs.** **Note: These bits are cleared to 0 only on power-on reset. The value of this register remains unchanged after all other resets.**

2.5.1.38 System Software Interrupt Request 1 Register (SSIR1)

The SSIR1 register, shown in Figure 2-41 and described in Table 2-55, is used for software interrupt generation.

NOTE: This register is mirrored at offset FCh for compatibility reasons.

Figure 2-41. System Software Interrupt Request 1 Register (SSIR1) (offset = B0h)

31	16
Reserved	
R-0	

15	8	7	0
SSKEY1		SSDATA1	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-55. System Software Interrupt Request 1 Register (SSIR1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSKEY1	0-FFh	System software interrupt request key. A 075h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY1 field can be written into only if the write data matches the key (75h). The SSDATA1 field can only be written into if the write data into this field, the SSKEY1 field, matches the key (75h).
7-0	SSDATA1	0-FFh	System software interrupt data. These bits contain user read/write register bits. They may be used by the application software as different entry points for the interrupt routine. The SSDATA1 field cannot be written into unless the write data into the SSKEY1 field matches the key (75h); therefore, byte writes cannot be performed on the SSDATA1 field.

2.5.1.39 System Software Interrupt Request 2 Register (SSIR2)

The SSIR2 register, shown in Figure 2-42 and described in Table 2-56, is used for software interrupt generation.

Figure 2-42. System Software Interrupt Request 2 Register (SSIR2) (offset = B4h)

31	16
Reserved	
R-0	

15	8	7	0
SSKEY2		SSDATA2	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-56. System Software Interrupt Request 2 Register (SSIR2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSKEY2	0-FFh	System software interrupt2 request key. A 84h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY2 field can be written into only if the write data matches the key (84h). The SSDATA2 field can only be written into if the write data into this field, the SSKEY2 field, matches the key (84h).
7-0	SSDATA2	0-FFh	System software interrupt data. These bits contain user read/write register bits. They may be used by the application software as different entry points for the interrupt routine. The SSDATA2 field cannot be written into unless the write data into the SSKEY2 field matches the key (84h); therefore, byte writes cannot be performed on the SSDATA2 field.

2.5.1.40 System Software Interrupt Request 3 Register (SSIR3)

The SSIR3 register, shown in Figure 2-43 and described in Table 2-57, is used for software interrupt generation.

Figure 2-43. System Software Interrupt Request 3 Register (SSIR3) (offset = B8h)

31	16
Reserved	
R-0	

15	8	7	0
SSKEY3		SSDATA3	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-57. System Software Interrupt Request 3 Register (SSIR3) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSKEY3	0-FFh	System software interrupt request key. A 93h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY3 field can be written into only if the write data matches the key (93h). The SSDATA3 field can only be written into if the write data into this field, the SSKEY3 field, matches the key (93h).
7-0	SSDATA3	0-FFh	System software interrupt data. These bits contain user read/write register bits. They may be used by the application software as different entry points for the interrupt routine. The SSDATA3 field cannot be written into unless the write data into the SSKEY3 field matches the key (93h); therefore, byte writes cannot be performed on the SSDATA3 field.

2.5.1.41 System Software Interrupt Request 4 Register (SSIR4)

The SSIR4 register, shown in Figure 2-44 and described in Table 2-58, is used for software interrupt generation.

Figure 2-44. System Software Interrupt Request 4 Register (SSIR4) (offset = BCh)

31	16
Reserved	
R-0	

15	8	7	0
SSKEY4		SSDATA4	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-58. System Software Interrupt Request 4 Register (SSIR4) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSKEY4	0-FFh	System software interrupt2 request key. A A2h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY4 field can be written into only if the write data matches the key (A2h). The SSDATA4 field can only be written into if the write data into this field, the SSKEY4 field, matches the key (A2h).
7-0	SSDATA4	0-FFh	System software interrupt data. These bits contain user read/write register bits. They may be used by the application software as different entry points for the interrupt routine. The SSDATA4 field cannot be written into unless the write data into the SSKEY4 field matches the key (A2h); therefore, byte writes cannot be performed on the SSDATA4 field.

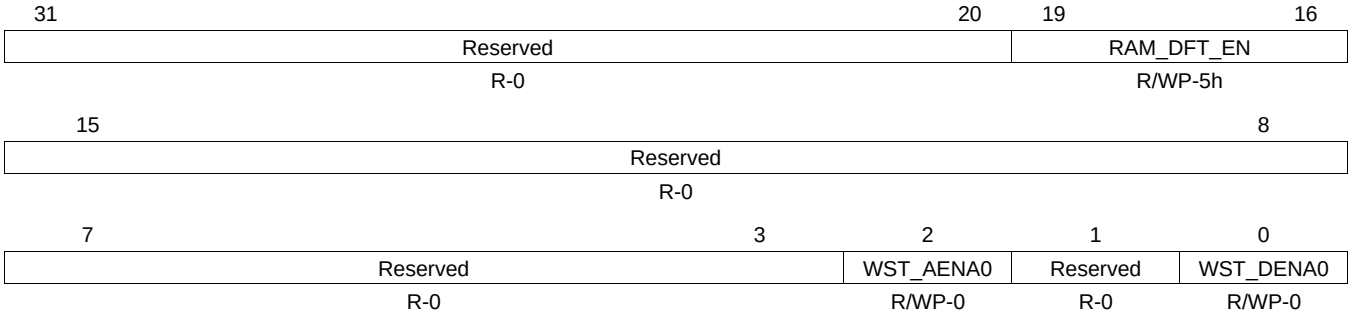
2.5.1.42 RAM Control Register (RAMGCR)

The RAMGCR register, shown in Figure 2-45 and described in Table 2-59, is used to configure eSRAM data and address wait states.

NOTE: The RAM_DFT_EN bits are for TI internal use only.

The contents of the RAM_DFT_EN field should not be changed.

Figure 2-45. RAM Control Register (RAMGCR) (offset = C0h)



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-59. RAM Control Register (RAMGCR) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	RAM_DFT_EN		Functional mode RAM DFT (Design For Test) port enable key.
			Note: For TI internal use only.
		Ah	RAM DFT port is enabled.
		All other values	RAM DFT port is disabled.
			Note: It is recommended that a value of 5h be used to disable the RAM DFT port. This value will give maximum protection from a bit flip inducing event that would inadvertently enable the controller.
15-3	Reserved	0	Reads return 0. Writes have no effect.
2	WST_AENA0		eSRAM data phase wait state enable bit.
		0	The default address setup time for eSRAM0 is used.
		1	The eSRAM address setup time is increased by one HCLK cycle.
1	Reserved	0	Reads return 0. Writes have no effect.
0	WST_DENA0		eSRAM data phase wait state enable bit.
		0	There are no wait states for eSRAM during the data phase.
		1	The eSRAM data phase setup time is increased by one HCLK cycle.

2.5.1.43 Bus Matrix Module Control Register 1 (BMMCR1)

The BMMCR1 register, shown in Figure 2-46 and described in Table 2-60, allows RAM and Program (Flash) memory addresses to be swapped.

Figure 2-46. Bus Matrix Module Control Register 1 (BMMCR) (offset = C4h)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		MEMSW	
R-0		R/WP-Ah	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-60. Bus Matrix Module Control Register 1 (BMMCR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MEMSW		Memory swap key. **Note: A CPU reset must be issued after the memory swap key has been changed for the memory swap to occur. A CPU reset can be initiated by changing the state of the CPU RESET bit in the CPURSTCR register.**
		5h	Swapped memory-map: eSRAM starts at address 0. Program memory (Flash) starts at address 800 0000h.
		Ah	Default memory-map: Program memory (Flash) starts at address 0. eSRAM starts at address 800 0000h.
		All other values	The device memory-map is unchanged.

2.5.1.44 CPU Reset Control Register (CPURSTCR)

The CPURSTCR register shown in Figure 2-47 and described in Table 2-61 allows a reset to the Cortex-R4 CPU to be generated.

NOTE: The register bits in CPURSTCR are designated as high-integrity bits and have been implemented with error-correcting logic such that each bit, although read and written as a single bit, is actually a multi-bit key with error correction capability. As such, single-bit flips within the “key” can be corrected allowing protection of the system as a whole. An error detected is signaled to the ESM module.

Figure 2-47. CPU Reset Control Register (CPURSTCR) (offset = CCh)

31	16
Reserved	
R-0	

15	1	0
Reserved		CPU_RESET
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-61. CPU Reset Control Register (CPURSTGCR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	CPU_RESET	0-1	CPU Reset. Only the CPU is reset whenever this bit is toggled. There is no system reset.

2.5.1.45 Clock Control Register (CLKCNTL)

The CLKCNTL register, shown in Figure 2-48 and described in Table 2-62, controls peripheral reset and the peripheral clock divide ratios.

NOTE: VCLK and VCLK2 clock ratio restrictions.

The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency.

In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously. When increasing the frequency (decreasing the divider), first change the VCLK2R field and then change the VCLKR field. When reducing the frequency (increasing the divider), first change the VCLKR field and then change the VCLK2R field.

You should do a read-back between the two writes. This assures that there are enough clock cycles between the two writes.

Figure 2-48. Clock Control Register (CLKCNTRL) (offset = D0h)

31	28	27	24	23	20	19	16
Reserved		VCLK2R		Reserved		VCLKR	
R-0		R/WP-1h		R-0		R/WP-1h	

15	9	8	7	0
Reserved		PENA	Reserved	
R-0		R/WP-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-62. Clock Control Register (CLKCNTRL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	VCLK2R		VBUS clock2 ratio.
			Note: The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously.
		0	The VCLK2 speed is HCLK divided by 1.
		1	The VCLK2 speed is HCLK divided by 2.
		2h-Fh	Reserved
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	VCLKR		VBUS clock ratio.
			Note: The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously.
		0	The VCLK speed is HCLK divided by 1.
		1	The VCLK speed is HCLK divided by 2.
		2h-Fh	Reserved
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	PENA		Peripheral enable bit. The application must set this bit before accessing any peripheral
		0	The global peripheral/peripheral memory frames are in reset.
		1	All peripheral/peripheral memory frames are out of reset.
7-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.1.46 ECP Control Register (ECPCNTL)

The ECP register, shown in Figure 2-49 and described in Table 2-63, configures the ECLK pin in functional mode.

NOTE: ECLK Functional mode configuration.

The ECLK pin must be placed into Functional mode by setting the ECPCLKFUN bit to 1 in the SYSPC1 register before a clock source will be visible on the ECLK pin.

Figure 2-49. ECP Control Register (ECPCNTL) (offset = D4h)

31	25	24	23	22	18	17	16
Reserved		ECPSSEL	ECPCOS	Reserved		ECPINSEL	
R-0		R/W-0	R/W-0	R-0		R/W-0	

15	0
ECPDIV	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2-63. ECP Control Register (ECPCNTL) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	ECPSSEL		This bit allows the selection between VCLK and OSCIN as the clock source for ECLK. **Note: Other ECLK clock sources are available for debug purposes by configuring the CLKTEST register.**
		0	VCLK is selected as the ECP clock source.
		1	OSCIN is selected as the ECP clock source.
23	ECPCOS		ECP continue on suspend. **Note: Suspend mode is entered while performing certain JTAG debugging operations.**
		0	ECLK output is disabled in suspend mode. ECLK output will be shut off and will not be seen on the I/O pin of the device.
		1	ECLK output is not disabled in suspend mode. ECLK output will not be shut off and will be seen on the I/O pin of the device.
22-18	Reserved	0	Reads return 0. Writes have no effect.
17-16	ECPINSEL		Select ECP input clock source.
		0	Tied Low
		1h	HCLK
		2h	External clock
		3h	Tied Low
15-0	ECPDIV	0-FFFFh	ECP divider value. The value of ECPDIV bits determine the external clock (ECP clock) frequency as a ratio of VBUS clock or OSCIN as shown in the formula: $ECLK = \frac{VCLK\,or\,OSCIN}{(ECPDIV + 1)}$

2.5.1.47 DEV Parity Control Register 1 (DEVCR1)

This register is shown in Figure 2-50 and described in Table 2-64.

Figure 2-50. DEV Parity Control Register 1 (DEVCR1) (offset = DCh)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		DEVPARSEL	
R-0		R/WP-Ah	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-64. DEV Parity Control Register 1 (DEVCR1) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	DEVPARSEL		Device parity select bit key. **Note: After an odd (DEVPARSEL = 5h) or even (DEVPARSEL = Ah) scheme is programmed into the DEVPARSEL register, any one bit change can be detected and will retain its programmed scheme. More than one bit changes in DEVPARSEL will cause a default to odd parity scheme.**
		5h	The device parity is even.
		Ah	The device parity is odd.

2.5.1.48 System Exception Control Register (SYSECR)

The SYSECR register, shown in Figure 2-51 and described in Table 2-65, is used to generate a software reset.

NOTE: The register bits in SYSECR are designated as high-integrity bits and have been implemented with error-correcting logic such that each bit, although read and written as a single bit, is actually a multi-bit key with error correction capability. As such, single-bit flips within the “key” can be corrected allowing protection of the system as a whole. An error detected is signaled to the ESM module.

Figure 2-51. System Exception Control Register (SYSECR) (offset = E0h)

31	16
Reserved	
R-0	

15	14	13	0
RESET1	RESET0	Reserved	
R/WP-0	R/WP-1	R-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

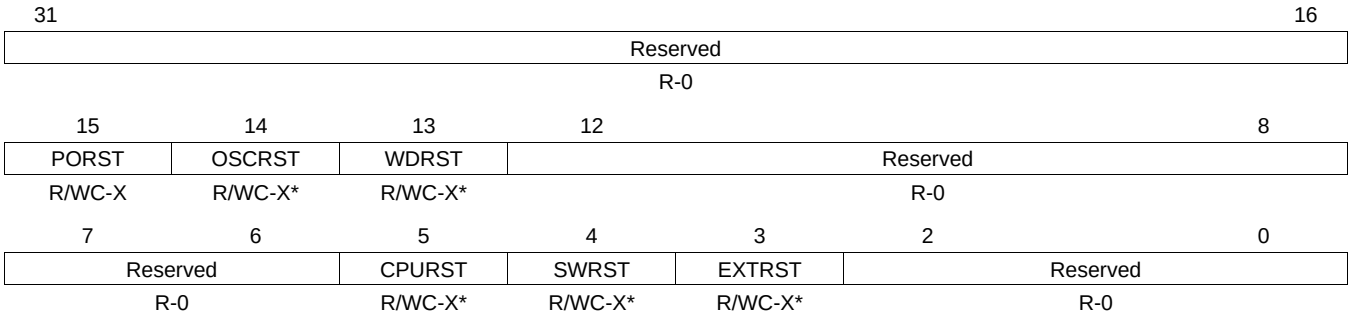
Table 2-65. System Exception Control Register (SYSECR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-14	RESET		Software reset bits. Setting RESET1 or clearing RESET0 causes a system software reset.
		1h	No reset will occur.
		0, 2h-3h	A global system reset will occur.
13-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.1.49 System Exception Status Register (SYSESR)

The SYSESR register, shown in Figure 2-52 and described in Table 2-66, shows the source for different resets encountered. Previous reset source status bits are not automatically cleared if new resets occur. After reading this register, the software should clear any flags that are set so that the source of future resets can be determined. Any bit in this register can be cleared by writing a 1 to the bit.

Figure 2-52. System Exception Status Register (SYSESR) (offset = E4h)



LEGEND: R/W = Read/Write; R = Read only; C= Clear; X = value unchanged after reset; X* = 0 after PORST but unchanged after other resets; -n = value after reset; *-n* = value after reset

Table 2-66. System Exception Status Register (SYSESR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15	PORST		Power-on reset. This bit is set when a power-on reset occurs, either internally asserted by the VMON or externally asserted by the nPORRST pin.
		0	No power-on reset has occurred since this bit was last cleared.
		1	A reset was caused by a power-on reset. (This bit should be cleared after being read so that subsequent resets can be properly identified as not being power-on resets.)
14	OSCRST		Reset caused by an oscillator failure or PLL cycle slip. This bit is set when a reset is caused by an oscillator failure or PLL slip.
			Note: The action taken when an oscillator failure or PLL slip is detected must configured in the PLLCTL1 register.
		0	No reset has occurred due to an oscillator failure or a PLL cycle slip.
		1	A reset was caused by an oscillator failure or a PLL cycle slip.
13	WDRST		Watchdog reset flag. This bit is set when the last reset was caused by the digital watchdog (DWD).
			During debugging, the ICEPICK logic implemented on the microcontroller also allows a system reset to be generated via the debug logic (DBGRST). This DBGRST reset is also indicated on the WDRST bit of the SYSESR. This flag can also be set via a reset driven by ICEPICK.
		0	No reset has occurred because of the DWD.
		1	A reset was caused by the DWD.
12-6	Reserved	0	Reads return 0. Writes have no effect.
5	CPURST		CPU reset flag. This bit is set when the CPU is reset.
			Note: A CPU reset can be initiated by the CPU self-test controller (LBIST) or by changing the memory protection (MMU/MPU) configuration in CPURSTCR register.
		0	No CPU reset has occurred.
		1	A CPU reset occurred.
4	SWRST		Software reset flag. This bit is set when a software system reset has occurred.
			Note: A software system reset can be initiated by writing to the RESET bits in the SYSECR register.
		0	No software reset has occurred.
		1	A software reset occurred.

Table 2-66. System Exception Status Register (SYSESR) Field Descriptions (continued)

Bit	Field	Value	Description
3	EXTRST		External reset flag. This bit is set when a reset is caused by the external reset pin nRST or by any reset that also asserts the nRST pin (PORST, OSCRST, WDRST and SWRST).
		0	The external reset pin has not asserted a reset.
		1	A reset has been caused by the external reset pin.
2-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.1.50 System Test Abort Status Register (SYSTASR)

This register is shown in Figure 2-53 and described in Table 2-67.

Figure 2-53. System Test Abort Status Register (SYSTASR) (offset = E8h)

31	16
Reserved	
R-0	

15	5	4	0
Reserved		EFUSE_Abort	
R-0		R/WCP-X/0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; C = Clear; -X = Value unchanged after reset; -*n* = value after reset

Table 2-67. System Test Abort Status Register (SYSTASR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	EFUSE_Abort		Test Abort status flag. These bits are set when test abort occurred:
		0	Read: The last operation (if any) completed successfully. This is also the value that the error/status register is set to after reset.
		1h	Read: Controller times out because there is no last row sent from the FuseROM.
		2h	Read: The autoload machine was started, either through the SYS_INITZ signal from the system or the JTAG data register. In either case, the autoload machine did not find enough FuseROM data to fill the scan chain.
		3h	Read: The autoload machine was started, either through the SYS_INITZ signal from the system or the JTAG data register. In either case, the autoload machine starts the scan chain with a signature it expects to see after the scan chain is full. The autoload machine was able to fill the scan chain, but the wrong signature was returned.
		4h	Read: The autoload machine was started, either through the SYS_INITZ signal from the system or the JTAG data register. In either case, the autoload machine was not able or not allowed to complete its operation.
		All other values	Read: Reserved.
		1Fh	Write: These bits are cleared to 0.

2.5.1.51 Global Status Register (GLBSTAT)

The GLBSTAT register, shown in Figure 2-54 and described in Table 2-68, indicates the FMzPLL (PLL1) slip status and the oscillator fail status.

NOTE: PLL and OSC fail behavior

The device behavior after a PLL slip or an oscillator failure is configured in the PLLCTL1 register.

Figure 2-54. Global Status Register (GLBSTAT) [offset = ECh]

31	16
Reserved	
R-0	

15	10	9	8	7	1	0
Reserved		FBSLIP	RFSLIP	Reserved		OSCFAIL
R-0		R/W1C-n	R/W1C-n	R-0		R/W1C-n

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to Clear; -*n* = value after reset

Table 2-68. Global Status Register (GLBSTAT) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	FBSLIP		PLL over cycle slip detection. (cleared by nPORRST, maintains its previous value for all other resets)
		0	Read: No PLL over cycle slip has been detected. Write: The bit is unchanged.
		1	Read: A PLL over cycle slip has been detected. Write: The bit is cleared to 0.
8	RFSLIP		PLL under cycle slip detection. (cleared by nPORRST, maintains its previous value for all other resets)
		0	Read: No PLL under cycle slip has been detected. Write: The bit is unchanged.
		1	Read: A PLL under cycle slip has been detected. Write: The bit is cleared to 0.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	OSCFAIL		Oscillator fail flag bit. (cleared by nPORRST, maintains its previous value for all other resets)
		0	Read: No oscillator failure has been detected. Write: The bit is unchanged.
		1	Read: An oscillator failure has been detected. Write: The bit is cleared to 0.

2.5.1.52 Device Identification Register (DEVID)

The DEVID is a read-only register. It contains device-specific information that is hard-coded during device manufacture. For the initial silicon version, the device identification code value is 0x8048AD05. This register is shown in Figure 2-55 and described in Table 2-69.

Figure 2-55. Device Identification Register (DEVID) (offset = F0h)

31	30	17	16
CP15	UNIQUE_ID		TECH
R-K	R-K		R-K

15	13	12	11	10	9	8
TECH		IO_VOLTAGE	PERIPHERAL_PARITY	FLASH_ECC		RAM_ECC
R-K		R-K	R-K	R-K		R-K

7	3	2	0
VERSION		PLATFORM_ID	
R-K		R-K	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset; *-K* = constant value

Table 2-69. Device Identification Register (DEVID) Field Descriptions

Bit	Field	Value	Description
31	CP15		CP15 CPU. This bit indicates whether the CPU has a coprocessor 15 (CP15).
		0	The CPU has no CP15 present.
		1	The CPU has a CP15 present. The CPU ID can be read using the CP15 C0,C0,0 register.
30-17	UNIQUE_ID	0-3FFFh	Device ID. The device ID is unique by device configuration.
16-13	TECH		These bits define the process technology by which the device was manufactured.
		0	Device manufactured in the C05 process technology.
		1h	Device manufactured in the F05 process technology.
		2h	Device manufactured in the C035 process technology.
		3h	Device manufactured in the F035 process technology.
		4h	Device manufactured in the C021 process technology.
		5h	Device manufactured in the F021 process technology.
		6h-Fh	Reserved
12	IO_VOLTAGE		Input/output voltage. This bit defines the I/O voltage of the device.
		0	The I/O voltage is 3.3 V.
		1	The I/O voltage is 5 V.
11	PERIPHERAL_PARITY		Peripheral parity. This bit indicates whether or not peripheral memory parity is present.
		0	The peripheral memories have no parity.
		1	The peripheral memories have parity.
10-9	FLASH_ECC		These bits indicate which parity is present for the program memory.
		0	No memory protection is present.
		1h	The program memory (Flash) has single-bit parity.
		2h	The program memory (Flash) has ECC.
		3h	Reserved
8	RAM_ECC		RAM ECC. This bit indicates whether or not RAM memory ECC is present.
		0	The RAM memories do not have ECC.
		1	The RAM memories have ECC.
7-3	VERSION	0-1Fh	Version. These bits provide the revision of the device.
2-0	PLATFORM_ID	5h	The device is part of the TMS570 family. The TMS570 ID is always 5h.

2.5.1.53 Software Interrupt Vector Register (SSIVEC)

The SSIVEC register, shown in Figure 2-56 and described in Table 2-70, contains information about software interrupts.

Figure 2-56. Software Interrupt Vector Register (SSIVEC) (offset = F4h)

31	16
Reserved	
R-0	

15	8	7	0
SSIDATA		SSIVECT	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

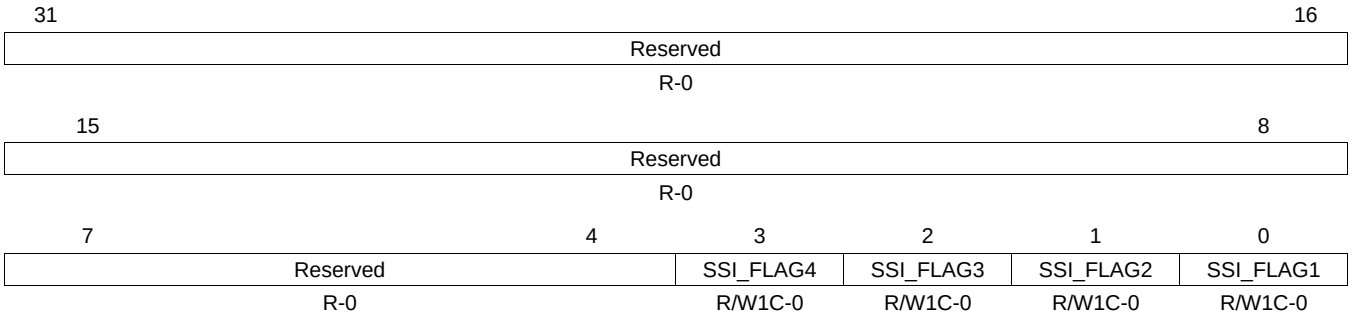
Table 2-70. Software Interrupt Vector Register (SSIVEC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSIDATA	0-FFh	System software interrupt data key. These bits contain the data key value of the source for the system software interrupt, which is indicated by the vector in the SSIVEC[7-0] field.
7-0	SSIVECT		These bits contain the source for the system software interrupt. **Note: A read from the SSIVECT bits clears the corresponding SSI_FLAG[4-1] bit in the SSIF register, corresponding to the source vector of the system software interrupt.** **Note: The SSIR[4-1] interrupt has the following priority order:** **SSIR1 has the highest priority.** **SSIR4 has the lowest priority.**
		0	No software interrupt is pending.
		1h	A software interrupt has been generated by writing the correct key value to The SSIR1 register.
		2h	A software interrupt has been generated by writing the correct key value to The SSIR2 register.
		3h	A software interrupt has been generated by writing the correct key value to The SSIR3 register.
		4h	A software interrupt has been generated by writing the correct key value to The SSIR4 register.
		5h-FFh	Reserved

2.5.1.54 System Software Interrupt Flag Register (SSIF)

The SSIF register, shown in Figure 2-57 and described in Table 2-71, contains software interrupt flag status information.

Figure 2-57. System Software Interrupt Flag Register (SSIF) (offset = F8h)



LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 2-71. System Software Interrupt Flag Register (SSIF) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	SSI_FLAG[4-1]		System software interrupt flag[4-1]. This flag is set when the correct SSKEY is written to the SSIR register[4-1]. **Note: A read from the SSIVEC register clears the corresponding SSI_FLAG[4-1] bit in the SSIF, corresponding to the source vector of the system software interrupt.**
		0	Read: No IRQ/FIQ interrupt was generated since the bit was last cleared. Write: The bit is unchanged.
		1	Read: An IRQ/FIQ interrupt was generated. Write: The bit is cleared to 0.

2.5.2 Secondary System Control Registers (SYS2)

This section describes the secondary frame of system registers. The start address of the secondary system module frame is FFFF E100. The registers support 32-, 16-, and 8-bit writes. The offset is relative to the system module frame start address.

Table 2-72 lists the secondary system control registers.

NOTE: All additional registers in the secondary system frame are reserved.

Table 2-72. Secondary System Control Registers

Offset	Acronym	Register Description	Section
08h	STCCLKDIV	CPU Logic BIST Clock Divider	Section 2.5.2.1
70h	CLKSLIP	Clock Slip Register	Section 2.5.2.2
ECh	EFC_CTLREG	EFUSE Controller Control Register	Section 2.5.2.3
F0h	DIEIDL_REG0	Die Identification Register Lower Word	Section 2.5.2.4
F4h	DIEIDH_REG1	Die Identification Register Upper Word	Section 2.5.2.5
F8h	DIEIDL_REG2	Die Identification Register Lower Word	Section 2.5.2.6
FCh	DIEIDH_REG3	Die Identification Register Upper Word	Section 2.5.2.7

2.5.2.1 CPU Logic Bist Clock Divider (STCLKDIV)

This register is shown in Figure 2-58 and described in Table 2-73.

Figure 2-58. CPU Logic BIST Clock Prescaler (STCLKDIV) (offset = 08h)

31 27	26 24	23 16
Reserved	CLKDIV	Reserved
R-0	R/WP-0	R-0

15 0
Reserved
R-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-73. CPU Logic BIST Clock Prescaler (STCLKDIV) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26-24	CLKDIV		Clock divider/prescaler for CPU clock (GCLK) during logic BIST
		0	STCCLK = HCLK/1
		1h	STCCLK = HCLK/2
		2h	STCCLK = HCLK/3
		:	:
		7h	STCCLK = HCLK/8
23-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.2.2 Clock Slip Register (CLKSLIP)

This register is shown in Figure 2-59 and described in Table 2-74.

Figure 2-59. Clock Slip Register (CLKSLIP) (offset = 70h)

31	16
Reserved	
R-0	

15	14	13	8	7	4	3	0
Reserved		PLL1_SLIP_FILTER_COUNT		Reserved		PLL1_SLIP_FILTER_KEY	
R-0		R/WP-0		R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-74. Clock Slip Register (CLKSLIP) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reads return 0. Writes have no effect.
13-8	PLL1_SLIP_FILTER_COUNT		PLL SLIP down counter programmed value. Count is on 10M clock.
			On reset, counter value is 0. Counter must be programmed to a non-zero value and enabled for the filtering to be enabled.
		0	Filtering is disabled.
		1h	Filtering is enabled. Every slip is recognized.
		2h	Filtering is enabled. The slip must be at least 2 HF LPO cycles wide in order to be recognized as a slip.
		:	:
		3Fh	Filtering is enabled. The slip must be at least 63 HF LPO cycles wide in order to be recognized as a slip.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PLL1_SLIP_FILTER_KEY		Enable the PLL SLIP filtering.
		5h	On reset, the PLL SLIP filter is disabled and the PLL SLIP passes through.
		Fh	This is an unsupported value. You should avoid writing this value to this bit field.
		All other values	PLL SLIP filtering is enabled. Recommended to program Ah in this bit field. Enabling of the PLL SLIP occurs when the KEY is programmed and a nonzero value is present in the COUNT field.

2.5.2.3 EFUSE Controller Control Register (EFC_CTLREG)

This register is shown in Figure 2-60 and described in Table 2-75.

Figure 2-60. EFUSE Controller Control Register (EFC_CTLREG) (offset = ECh)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		EFC_INSTR_WEN	
R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-75. EFUSE Controller Control Register (EFC_CTLREG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EFC_INSTR_WEN		Enable user write of 4 EFUSE controller instructions.
			SYS module generates the enable signal which will be tied to OCP_FROM_WRITE_DISABLE on efuse controller port
		Ah	Writing of instructions (Program, ProgramCRA, RunAutoload, and LoadFuseScanchain) to EFC is allowed enabled.
		All other values	Writing of instructions (Program, ProgramCRA, RunAutoload, and LoadFuseScanchain) in EFC registers is blocked.

2.5.2.4 Die Identification Register Lower Word (DIEIDL_REG0)

The DIEIDL_REG0 register is a duplicate of the DIEIDL register, see Section 2.5.1.29. The DIEIDL_REG0 register, shown in Figure 2-61 and described in Table 2-76, contains information about the die wafer number, and X, Y wafer coordinates.

Figure 2-61. Die Identification Register, Lower Word (DIEIDL_REG0) [offset = F0h]

31	24	23	16
WAFER #		Y WAFER COORDINATE	
R-D		R-D	

15	12	11	0
Y WAFER COORDINATE		X WAFER COORDINATE	
R-D		R-D	

LEGEND: R = Read only; -*n* = value after reset; -D = device specific

Table 2-76. Die Identification Register, Lower Word (DIEIDL_REG0) Field Descriptions

Bit	Field	Description
31-24	WAFER #	These read-only bits contain the wafer number of the device.
23-12	Y WAFER COORDINATE	These read-only bits contain the Y wafer coordinate of the device.
11-0	X WAFER COORDINATE	These read-only bits contain the X wafer coordinate of the device.

NOTE: Die Identification Information

The die identification information will vary from unit to unit. This information is programmed by TI as part of the initial device test procedure.

2.5.2.5 Die Identification Register Upper Word (DIEIDH_REG1)

The DIEIDH_REG1 register is a duplicate of the DIEIDH register, see Section 2.5.1.30. The DIEIDH_REG1 register, shown in Figure 2-62 and described in Table 2-77, contains information about the die lot number.

Figure 2-62. Die Identification Register, Upper Word (DIEIDH_REG1) [offset = F4h]

31	24	23	16
Reserved		LOT #	
R-0		R-D	

15	0
LOT #	
R-D	

LEGEND: R/W = Read/Write; R = Read only; D = Value is device dependent; -*n* = value after reset

Table 2-77. Die Identification Register, Upper Word (DIEIDH_REG1) Field Descriptions

Bit	Field	Description
31-24	Reserved	Reserved for TI use. Writes have no effect.
23-0	LOT #	This read-only register contains the device lot number.

NOTE: Die Identification Information

The die identification information will vary from unit to unit. This information is programmed by TI as part of the initial device test procedure.

2.5.2.6 Die Identification Register Lower Word (DIEIDL_REG2)

This register is shown in Figure 2-63 and described in Table 2-78.

Figure 2-63. Die Identification Register, Lower Word (DIEIDL_REG2) [offset = F8h]

31	0
DIEIDL2	
R-X	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; -X = Value unchanged after reset

Table 2-78. Die Identification Register, Lower Word (DIEIDL_REG2) Field Descriptions

Bit	Field	Value	Description
31-0	DIEIDL2(95-64)	0-FFFF FFFFh	This read-only register contains the lower word (95:64) of the die ID information. The contents of this register is reserved.

2.5.2.7 Die Identification Register Upper Word (DIEIDH_REG3)

This register is shown in Figure 2-64 and described in Table 2-79.

Figure 2-64. Die Identification Register, Upper Word (DIEIDH_REG3) [offset = FCh]

31	0
DIEIDH2	
R-X	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; -X = Value unchanged after reset

Table 2-79. Die Identification Register, Upper Word (DIEIDH_REG3) Field Descriptions

Bit	Field	Value	Description
31-0	DIEIDH2(127-96)	0-FFFF FFFFh	This read-only register contains the upper word (127:97) of the die ID information. The contents of this register is reserved.

2.5.3 Peripheral Central Resource (PCR) Control Registers

This section describes the Peripheral Central Resource (PCR) control registers. The start address of the PCR register frame is FFFF E000h. Table 2-80 lists the registers in the PCR, which are used to configure protection to the peripherals in PCS and PS regions. Not all chip selects exist on this device.

Table 2-80. Peripheral Central Resource Control Registers

Offset	Acronym	Register Description	Section
00h	PMPROTSET0	Peripheral Memory Protection Set Register 0	Section 2.5.3.1
04h	PMPROTSET1	Peripheral Memory Protection Set Register 1	Section 2.5.3.2
10h	PMPROTCLR0	Peripheral Memory Protection Clear Register 0	Section 2.5.3.3
14h	PMPROTCLR1	Peripheral Memory Protection Clear Register 1	Section 2.5.3.4
20h	PPROTSET0	Peripheral Protection Set Register 0	Section 2.5.3.5
24h	PPROTSET1	Peripheral Protection Set Register 1	Section 2.5.3.6
28h	PPROTSET2	Peripheral Protection Set Register 2	Section 2.5.3.7
2Ch	PPROTSET3	Peripheral Protection Set Register 3	Section 2.5.3.8
40h	PPROTCLR0	Peripheral Protection Clear Register 0	Section 2.5.3.9
44h	PPROTCLR1	Peripheral Protection Clear Register 1	Section 2.5.3.10
48h	PPROTCLR2	Peripheral Protection Clear Register 2	Section 2.5.3.11
4Ch	PPROTCLR3	Peripheral Protection Clear Register 3	Section 2.5.3.12
60h	PCSPWRDWNSET0	Peripheral Memory Power-Down Set Register 0	Section 2.5.3.13
64h	PCSPWRDWNSET1	Peripheral Memory Power-Down Set Register 1	Section 2.5.3.14
70h	PCSPWRDWNCLR0	Peripheral Memory Power-Down Clear Register 0	Section 2.5.3.15
74h	PCSPWRDWNCLR1	Peripheral Memory Power-Down Clear Register 1	Section 2.5.3.16
80h	PSPWRDWNSET0	Peripheral Power-Down Set Register 0	Section 2.5.3.17
84h	PSPWRDWNSET1	Peripheral Power-Down Set Register 1	Section 2.5.3.18
88h	PSPWRDWNSET2	Peripheral Power-Down Set Register 2	Section 2.5.3.19
8Ch	PSPWRDWNSET3	Peripheral Power-Down Set Register 3	Section 2.5.3.20
A0h	PSPWRDWNCLR0	Peripheral Power-Down Clear Register 0	Section 2.5.3.21
A4h	PSPWRDWNCLR1	Peripheral Power-Down Clear Register 1	Section 2.5.3.22
A8h	PSPWRDWNCLR2	Peripheral Power-Down Clear Register 2	Section 2.5.3.23
ACh	PSPWRDWNCLR3	Peripheral Power-Down Clear Register 3	Section 2.5.3.24

2.5.3.1 Peripheral Memory Protection Set Register 0 (PMPROTSET0)

This register is shown in Figure 2-65 and described in Table 2-81.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-65. Peripheral Memory Protection Set Register 0 (PMPROTSET0) (offset = 00h)

31	0
PCS[31-0]PROTSET	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-81. Peripheral Memory Protection Set Register 0 (PMPROTSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[31-0]PROTSET		Peripheral memory frame protection set.
		0	Read: The peripheral memory frame*n* can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral memory frame*n* can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PMPROTSET0 and PMPROTCLR0 registers is set to 1.

2.5.3.2 Peripheral Memory Protection Set Register 1 (PMPROTSET1)

This register is shown in Figure 2-66 and described in Table 2-82.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-66. Peripheral Memory Protection Set Register 1 (PMPROTSET1) (offset = 04h)

31	0
PCS[63-32]PROTSET	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-82. Peripheral Memory Protection Set Register 1 (PMPROTSET1) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[63-32]PROTSET		Peripheral memory frame protection set.
		0	Read: The peripheral memory frame*n* can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral memory frame*n* can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PMPROTSET1 and PMPROTCLR1 registers is set to 1.

2.5.3.3 Peripheral Memory Protection Clear Register 0 (PMPROTCLR0)

This register is shown in Figure 2-67 and described in Table 2-83.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-67. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) (offset = 10h)

31	0
PCS[31-0]PROTCLR	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-83. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[31-0]PROTCLR		Peripheral memory frame protection clear.
		0	Read: The peripheral memory frame*n* can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral memory frame*n* can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PMPROTSET0 and PMPROTCLR0 registers is cleared to 0.

2.5.3.4 Peripheral Memory Protection Clear Register 1 (PMPROTCLR1)

This register is shown in Figure 2-68 and described in Table 2-84.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-68. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) (offset = 14h)

31	0
PCS[63-32]PROTCLR	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-84. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[63-32]PROTCLR		Peripheral memory frame protection clear.
		0	Read: The peripheral memory frame*n* can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral memory frame*n* can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PMPROTSET1 and PMPROTCLR1 registers is cleared to 0.

2.5.3.5 Peripheral Protection Set Register 0 (PPROTSET0)

There is one bit for each quadrant for PS0 to PS7.

The following are the ways in which quadrants are used within a PS frame:

a. The slave uses all the four quadrants:

   Only the bit corresponding to the quadrant 0 of PSn is implemented. It protects the whole 1K-byte frame. The remaining three bits are not implemented.

b. The slave uses two quadrants:

   Each quadrant has to be in one of these groups: (Quad 0 and Quad 1) or (Quad 2 and Quad 3).

   For the group Quad0/Quad1, the bit quadrant 0 protects both quadrants 0 and 1. The bit quadrant 1 is not implemented.

   For the group Quad2/Quad3, the bit quadrant 2 protects both quadrants 2 and 3. The bit quadrant 3 is not implemented.

c. The slave uses only one quadrant:

   In this case, the bit, as specified in Table 2-85, protects the slave.

The arrangement is true for all the peripheral select (PS0 to PS31).

This register holds bits for PS0 to PS7 and is shown in Figure 2-69 and described in Table 2-85.

NOTE: Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-69. Peripheral Protection Set Register 0 (PPROTSET0) (offset = 20h)

31	0
PS[7-0]QUAD[3-0]PROTSET	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-85. Peripheral Protection Set Register 0 (PPROTSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0]PROTSET		Peripheral select quadrant protection set.
		0	Read: The peripheral select quadrant an be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PPROTSET0 and PPROTCLR0 registers is set to 1.

2.5.3.6 Peripheral Protection Set Register 1 (PPROTSET1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-70 and described in Table 2-86.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-70. Peripheral Protection Set Register 1 (PPROTSET1) (offset = 24h)

31	0
PS[15-8]QUAD[3-0]PROTSET	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-86. Peripheral Protection Set Register 1 (PPROTSET1) Field Descriptions

Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0]PROTSET		Peripheral select quadrant protection set.
		0	Read: The peripheral select quadrant can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PPROTSET1 and PPROTCLR1 registers is set to 1.

2.5.3.7 Peripheral Protection Set Register 2 (PPROTSET2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-71 and described in Table 2-87.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-71. Peripheral Protection Set Register 2 (PPROTSET2) (offset = 28h)

31	0
PS[23-16]QUAD[3-0]PROTSET	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-87. Peripheral Protection Set Register 2 (PPROTSET2) Field Descriptions

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0]PROTSET		Peripheral select quadrant protection set.
		0	Read: The peripheral select quadrant can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PPROTSET2 and PPROTCLR2 registers is set to 1.

2.5.3.8 Peripheral Protection Set Register 3 (PPROTSET3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-72 and described in Table 2-88.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-72. Peripheral Protection Set Register 3 (PPROTSET3) (offset = 2Ch)

31	0
PS[31-24]QUAD[3-0]PROTSET	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-88. Peripheral Protection Set Register 3 (PPROTSET3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0]PROTSET		Peripheral select quadrant protection set.
		0	Read: The peripheral select quadrant can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PPROTSET3 and PPROTCLR3 registers is set to 1.

2.5.3.9 Peripheral Protection Clear Register 0 (PPROTCLR0)

There is one bit for each quadrant for PS0 to PS7. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-73 and described in Table 2-89.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-73. Peripheral Protection Clear Register 0 (PPROTCLR0) (offset = 40h)

31	0
PS[7-0]QUAD[3-0]PROTCLR	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-89. Peripheral Protection Clear Register 0 (PPROTCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0]PROTCLR		Peripheral select quadrant protection clear.
		0	Read: The peripheral select quadrant can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PPROTSET0 and PPROTCLR0 registers is cleared to 0.

2.5.3.10 Peripheral Protection Clear Register 1 (PPROTCLR1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-74 and described in Table 2-90.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-74. Peripheral Protection Clear Register 1 (PPROTCLR1) (offset = 44h)

31	0
PS[15-8]QUAD[3-0]PROTCLR	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-90. Peripheral Protection Clear Register 1 (PPROTCLR1) Field Descriptions

Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0]PROTCLR		Peripheral select quadrant protection clear.
		0	Read: The peripheral select quadrant can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PPROTSET1 and PPROTCLR1 registers is cleared to 0.

2.5.3.11 Peripheral Protection Clear Register 2 (PPROTCLR2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-75 and described in Table 2-91.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-75. Peripheral Protection Clear Register 2 (PPROTCLR2) (offset = 48h)

31	0
PS[23-16]QUAD[3-0]PROTCLR	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-91. Peripheral Protection Clear Register 2 (PPROTCLR2) Field Descriptions

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0]PROTCLR		Peripheral select quadrant protection clear.
		0	Read: The peripheral select quadrant can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PPROTSET2 and PPROTCLR2 registers is cleared to 0.

2.5.3.12 Peripheral Protection Clear Register 3 (PPROTCLR3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-76 and described in Table 2-92.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-76. Peripheral Protection Clear Register 3 (PPROTCLR3) (offset = 4Ch)

31	0
PS[31-24]QUAD[3-0]PROTCLR	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-92. Peripheral Protection Clear Register 3 (PPROTCLR3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0]PROTCLR		Peripheral select quadrant protection clear.
		0	Read: The peripheral select quadrant can be written to and read from in both user and privileged modes. Write: The bit is unchanged.
		1	Read: The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. Write: The corresponding bit in PPROTSET3 and PPROTCLR3 registers is cleared to 0.

2.5.3.13 Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0)

Each bit corresponds to a bit at the same index in the PMPROT register in that they both relate to the same peripheral. This register is shown in Figure 2-77 and described in Table 2-93.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-77. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) (offset = 60h)

31	0
PCS[31-0]PWRDNSET	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-93. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[31-0]PWRDNSET		Peripheral memory clock power-down set.
		0	Read: The peripheral memory clock[31-0] is active. Write: The bit is unchanged.
		1	Read: The peripheral memory clock[31-0] is inactive. Write: The corresponding bit in the PCSPWRDWNSET0 and PCSPWRDWNCLR0 registers is set to 1.

2.5.3.14 Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1)

This register is shown in Figure 2-78 and described in Table 2-94.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-78. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) (offset = 64h)

31	0
PCS[63-32]PWRDNSET	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-94. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[63-32]PWRDNSET		Peripheral memory clock power-down set.
		0	Read: The peripheral memory clock[63-32] is active. Write: The bit is unchanged.
		1	Read: The peripheral memory clock[63-32] is inactive. Write: The corresponding bit in the PCSPWRDWNSET1 and PCSPWRDWNCLR1 registers is set to 1.

2.5.3.15 Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0)

This register is shown in Figure 2-79 and described in Table 2-95.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-79. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) (offset = 70h)

31	0
PCS[31-0]PWRDNCLR	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-95. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[31-0]PWRDNCLR		Peripheral memory clock power-down clear.
		0	Read: The peripheral memory clock[31-0] is active. Write: The bit is unchanged.
		1	Read: The peripheral memory clock[31-0] is inactive. Write: The corresponding bit in the PCSPWRDWNSET0 and PCSPWRDWNCLR0 registers is cleared to 0.

2.5.3.16 Peripheral Memory Power-Down Clear Register 1 (PCSPWRDWNCLR1)

This register is shown in Figure 2-80 and described in Table 2-96.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-80. Peripheral Memory Power-Down Clear Register 1 (PCSPWRDWNCLR1) (offset = 74h)

31	0
PCS[63-32]PWRDNCLR	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-96. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNCLR1) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[63-32]PWRDNCLR		Peripheral memory clock power-down clear.
		0	Read: The peripheral memory clock[63-32] is active. Write: The bit is unchanged.
		1	Read: The peripheral memory clock[63-32] is inactive. Write: The corresponding bit in the PCSPWRDWNSET1 and PCSPWRDWNCLR1 registers is cleared to 0.

2.5.3.17 Peripheral Power-Down Set Register 0 (PSPWRDWNSET0)

There is one bit for each quadrant for PS0 to PS7. Each bit of this register corresponds to the bit at the same index in the corresponding PPROT register in that they relate to the same peripheral. These bits are used to power down/power up the clock to the corresponding peripheral.

For every bit implemented in the PPROT register, there is one bit in the PSnPWRDWN register, except when two peripherals (both in PS area) share buses. In that case, only one Power-Down bit is implemented, at the position corresponding to that peripheral whose quadrant comes first (the lower numbered).

The ways in which quadrants can be used within a frame are identical to what is described under PPROTSET0, Section 2.5.3.5.

This arrangement is the same for bits of PS8 to PS31, presented in Section 2.5.3.18 - Section 2.5.3.24. This register holds bits for PS0 to PS7. This register is shown in Figure 2-81 and described in Table 2-97.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-81. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) (offset = 80h)

31	0
PS[7-0]QUAD[3-0]PWRDWNSET	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-97. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0]PWRDWNSET		Peripheral select quadrant clock power-down set.
		0	Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is set to 1.

2.5.3.18 Peripheral Power-Down Set Register 1 (PSPWRDWNSET1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-82 and described in Table 2-98.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-82. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) (offset = 84h)

31	0
PS[15-8]QUAD[3-0]PWRDWNSET	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-98. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) Field Descriptions

Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0]PWRDWNSET	0	Peripheral select quadrant clock power-down set. Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDWNSET1 and PSPWRDWNCLR1 registers is set to 1.

2.5.3.19 Peripheral Power-Down Set Register 2 (PSPWRDWNSET2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-83 and described in Table 2-99.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-83. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) (offset = 88h)

31	0
PS[23-16]QUAD[3-0]PWRDWNSET	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-99. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) Field Descriptions

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0]PWRDWNSET	0	Peripheral select quadrant clock power-down set. Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDWNSET2 and PSPWRDWNCLR2 registers is set to 1.

2.5.3.20 Peripheral Power-Down Set Register 3 (PSPWRDWNSET3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-84 and described in Table 2-100.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-84. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) (offset = 8Ch)

31	0
PS[31-24]QUAD[3-0]PWRDWNSET	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-100. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0]PWRDWNSET		Peripheral select quadrant clock power-down set.
		0	Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDWNSET3 and PSPWRDWNCLR3 registers is set to 1.

2.5.3.21 Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0)

There is one bit for each quadrant for PS0 to PS7. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-85 and described in Table 2-101.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-85. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) (offset = A0h)

31	0
PS[7-0]QUAD[3-0]PWRDWNCLR	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-101. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0]PWRDWNCLR		Peripheral select quadrant clock power-down clear.
		0	Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is cleared to 0.

2.5.3.22 Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-86 and described in Table 2-102.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-86. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) (offset = A4h)

31	0
PS[15-8]QUAD[3-0]PWRDWNCLR	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-102. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) Field Descriptions

Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0]PWRDWNCLR	0	Peripheral select quadrant clock power-down clear. Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDWNSET1 and PSPWRDWNCLR1 registers is cleared to 0.

2.5.3.23 Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-87 and described in Table 2-103.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-87. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) (offset = A8h)

31	0
PS[23-16]QUAD[3-0]PWRDWNCLR	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-103. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) Field Descriptions

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0]PWRDWNCLR	0	Peripheral select quadrant clock power-down clear. Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDWNSET2 and PSPWRDWNCLR2 registers is cleared to 0.

2.5.3.24 Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-88 and described in Table 2-104.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-88. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR) (offset = ACh)

31	0
PS[31-24]QUAD[3-0]PWRDWNCLR	
R/WP-1	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 2-104. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0]PWRDWNCLR		Peripheral select quadrant clock power-down clear.
		0	Read: The clock to the peripheral select quadrant is active. Write: The bit is unchanged.
		1	Read: The clock to the peripheral select quadrant is inactive. Write: The corresponding bit in PSPWRDWNSET3 and PSPWRDWNCLR3 registers is cleared to 0.

I/O Multiplexing and Control Module (IOMM)

This chapter describes the I/O Multiplexing and Control Module (IOMM).

Topic **Page**

3.1 Overview

This chapter describes the overall features of the module that controls the I/O multiplexing on the device. The mapping of control registers to multiplexing options is specified in Section 3.5.12.

3.2 Main Features of I/O Multiplexing Module (IOMM)

The IOMM contains memory-mapped registers (MMR) that control device-specific multiplexed functions. The safety and diagnostic features of the IOMM are:

- Kicker mechanism to protect the MMRs from accidental writes
- Error indication for access violations

The safety features of the IOMM are described in Section 3.4.

3.3 Control of Multiplexed Functions

Several functions are multiplexed on the device. The following sections describe the multiplexing scheme and its implementation.

3.3.1 Control of Multiplexed Outputs

The signal multiplexing controlled by each memory-mapped control register (PINMMRn) is described in Table 3-14. Each byte in the PINMMR controls the functionality output on a single terminal. Consider the following example in Figure 3-1 for the PINMMR1 control register.

Figure 3-1. PINMMR1 Control Register

31	26	25	24
Reserved		SPI2nCS[2]	GIOA[4]
R/WP-0		R/WP-0	R/WP-0
23	18	17	16
Reserved		SPI2nCS[3]	GIOA[3]
R/WP-0		R/WP-0	R/WP-1
15	10	9	8
Reserved		SPI3nCS[1]	GIOA[2]
R/WP-0		R/WP-0	R/WP-1
7	2	1	0
Reserved		SPI3nCS[2]	GIOA[1]
R/WP-0		R/WP-0	R/WP-1

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

- Consider the multiplexing controlled by PINMMR1[15–8]. These bits control the multiplexing between the GIOA[2] and SPI3nCS[1] for this device. The default function is GIOA[2]. This is indicated by bit 8 of the PINMMR1 register being set to 1.
- If the application wants an SPI3nCS[1] signal, then bit 8 of PINMMR1 must be cleared to 0 and bit 9 must be set to 1.
- Each feature of the output function is determined by the function selected to be output on a terminal.

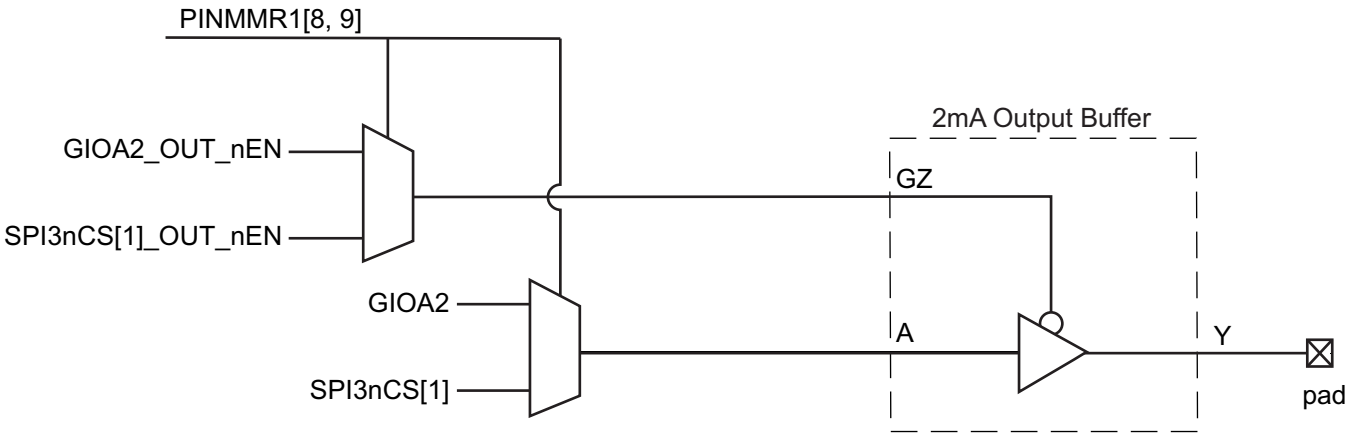
Figure 3-2 shows the multiplexing between the output functions for the pad. This terminal uses a 2mA output buffer.

For example, the pad is driven by an output buffer with an 2mA drive strength. This output buffer has the following signals: A (signal to be output), and GZ (output enable). Each of these signals is an output of a multiplexor that allows the selected function to control all available features of the output buffer.

The PINMMR control registers are used to implement a one-hot encoding scheme for selecting the multiplexed function:

- For example, at least one of bit 8 or bit 9 must be set.
- If the application clears both bits 8 and 9, then the default function (GIOA[2]) will be selected for output on the pad.
- If the application sets both bits 8 and 9, then the default function (GIOA[2]) will be selected for output on the pad.
- If the application sets one or more reserved bit(s) within the byte 15–8, then the default function (GIOA[2]) will be selected for output on the pad.

Figure 3-2. Output Multiplexing Example



3.3.2 Inputs

The input signals are broadcast to all modules hooked up to a terminal. The application must ensure that modules that are not being used in the application do not react to a change on their input functions. As in the example cited above, if the output function is selected as SPI3nCS[1], an output toggle will be viewable in the GIOA input register for the GIOA[2] input. Like wise, if GIOA[2] is selected on pin 5 and configured as an input, this input signal will also be seen in the SPI3 module nCS[1] bit location even though it was not selected to be active on pin 5.

3.3.3 Control of Special Multiplexed Options

Special controls are implemented to affect particular functions on this device. These controls are described in this section.

3.3.3.1 eQEPA Input

- When PINMMR8[0] = 1, the eQEPA input is double-synchronized using VCLK.
- When PINMMR8[0] = 0 and PINMMR8[1] = 1, the eQEPA input is double-synchronized and then qualified through a fixed 6-bit counter using VCLK.
- PINMMR8[0] = 0 and PINMMR8[1] = 0 is an invalid combination and behavior defaults to PINMMR8[0] = 1.

3.3.3.2 eQEPB Input

- When PINMMR8[8] = 1, the eQEPB input is double-synchronized using VCLK.
- When PINMMR8[8] = 0 and PINMMR8[9] = 1, the eQEPB input is double-synchronized and then qualified through a fixed 6-bit counter using VCLK.
- PINMMR8[8] = 0 and PINMMR8[9] = 0 is an invalid combination and behavior defaults to PINMMR8[8] = 1.

3.3.3.3 eQEPI Input

- When PINMMR8[16] = 1, the eQEPI input is double-synchronized using VCLK.
- When PINMMR8[16] = 0 and PINMMR8[17] = 1, the eQEPI input is double-synchronized and then qualified through a fixed 6-bit counter using VCLK.
- PINMMR8[16] = 0 and PINMMR8[17] = 0 is an invalid combination and behavior defaults to PINMMR8[16] = 1.

3.3.3.4 eQEPS Input

- When PINMMR8[24] = 1, the eQEPS input is double-synchronized using VCLK.
- When PINMMR8[24] = 0 and PINMMR8[25] = 1, the eQEPS input is double-synchronized and then qualified through a fixed 6-bit counter using VCLK.
- PINMMR8[24] = 0 and PINMMR8[25] = 0 is an invalid combination and behavior defaults to PINMMR8[24] = 1.

3.3.3.5 N2HET PIN_nDISABLE Input Port

- When PINMMR9[0] = 1, GIOA[5] is connected directly to N2HET PIN_nDISABLE input of the N2HET module.
- When PINMMR9[0] = 0 and PINMMR9[1] = 1, EQEPERR is inverted and double-synchronized using VCLK before connecting directly to the N2HET PIN_nDISABLE input of the N2HET module.
- PINMMR9[0] = 0 and PINMMR9[1] = 0 is an invalid combination and behavior defaults to PINMMR9[0] = 1.

3.4 Safety Features

The IOMM supports certain safety functions that are designed to prevent unintentional changes to the I/O multiplexing configuration. These are described in the following sections.

3.4.1 Locking Mechanism for Memory-Mapped Registers

The IOMM contains a mechanism to prevent any spurious writes from changing any of the PINMMR values. The PINMMRs are locked by default and after any system reset. None of the PINMMRs can be written under this condition. The application can read any of the IOMM registers regardless of the state of the locking mechanism.

- ***Enabling Write Access to the PINMMRs***

To enable write access to the PINMMRs, the CPU must write 0x83e70b13 to the kick0 register followed by a write of 0x95a4f1e0 to the kick1 register.

- ***Disabling Write Access to the PINMMRs***

It is recommended to disable write access to the PINMMRs once the I/O multiplexing configuration is completed. This can be done by:

- writing any other data value to either of the kick registers, or
- restarting the unlock sequence by writing 0x83e70b13 to the kick0 register

NOTE: No Error On Write to Locked PINMMRs

There is no error response on any write accesses to the PINMMRs when write access is disabled. None of the PINMMRs change state due to this write.

3.4.2 Error Conditions

The IOMM generates one error signal that is mapped to the Error Signaling Module's Group 1, channel 37. This error signal is generated under either of the following two conditions:

- Address Error – occurs when there is a read or a write access to an un-implemented memory location within the IOMM register frame.
- Protection Error – occurs when the CPU writes to an IOMM register while not in a privileged mode of operation.

3.5 IOMM Registers

Table 3-1 shows a summary of the control registers in the IOMM. The address offset is specified from the base address of FFFF EA00h.

Table 3-1. IOMM Register Summary

Offset	Acronym	Register Description	Section
00h	REVISION_REG	Revision Register	Section 3.5.1
20h	BOOT_REG	Boot Mode Register	Section 3.5.2
38h	KICK_REG0	Kicker Register 0	Section 3.5.3
3Ch	KICK_REG1	Kicker Register 1	Section 3.5.4
E0h	ERR_RAW_STATUS_REG	Error Raw Status / Set Register	Section 3.5.5
E4h	ERR_ENABLED_STATUS_REG	Error Enabled Status / Clear Register	Section 3.5.6
E8h	ERR_ENABLE_REG	Error Signaling Enable Register	Section 3.5.7
ECh	ERR_ENABLE_CLR_REG	Error Signaling Enable Clear Register	Section 3.5.8
F4h	FAULT_ADDRESS_REG	Fault Address Register	Section 3.5.9
F8h	FAULT_STATUS_REG	Fault Status Register	Section 3.5.10
FCh	FAULT_CLEAR_REG	Fault Clear Register	Section 3.5.11
110h-188h	PINMMRn	Pin Multiplexing Control Registers	Section 3.5.12

3.5.1 REVISION_REG: Revision Register

Figure 3-3. REVISION_REG: Revision Register (Offset = 00h)

31	30	29	28	27	16
REV SCHEME		Reserved		REV MODULE	
R-1		R-0		R-E84h	

15	11	10	8	7	6	5	0
REV RTL		REV MAJOR		REV CUSTOM		REV MINOR	
R-0		R-1		R-0		R-2h	

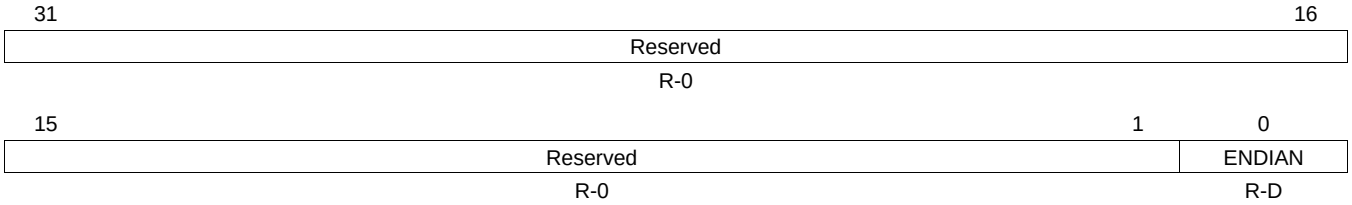
LEGEND: R = Read only; -*n* = value after reset

Table 3-2. Revision Register Field Descriptions

Bit	Field	Value	Description
31-30	REV SCHEME	1	Revision Scheme
29-28	Reserved	0	Reads return zeros, writes have no effect.
27-16	REV MODULE	E84h	Module Id
15-11	REV RTL	0	RTL Revision
10-8	REV MAJOR	1	Major Revision
7-6	REV CUSTOM	0	Custom Revision
5-0	REV MINOR	2h	Minor Revision

3.5.2 BOOT_REG: Boot Mode Register

Figure 3-4. BOOT_REG: Boot Mode Register (Offset = 20h)



LEGEND: Read only; -*n* = value after reset; D = value read is determined by external configuration

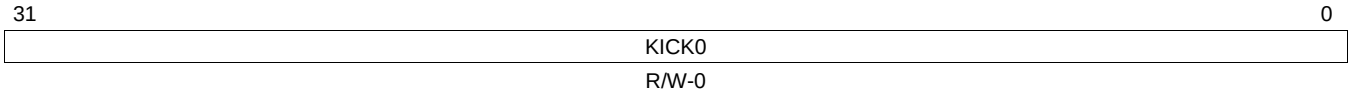
Table 3-3. Boot Mode Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	ENDIAN		Device endianness.
		0	Device is configured in little-endian mode.
		1	Device is configured in big-endian mode.

3.5.3 KICK_REG0: Kicker Register 0

This register forms the first part of the unlock sequence for being able to update the I/O multiplexing control registers (PINMMRnn).

Figure 3-5. KICK_REG0: Kicker Register 0 (Offset = 38h)



LEGEND: R/W = Read/Write; -*n* = value after reset

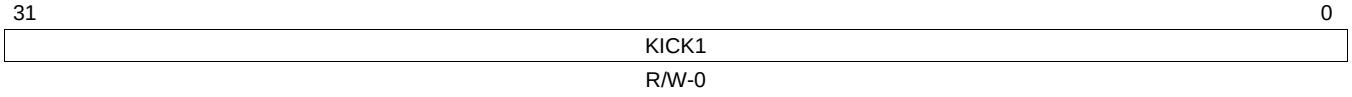
Table 3-4. Kicker Register 0 Field Descriptions

Bit	Field	Value	Description
31-0	KICK0	0	Kicker 0 Register. The value 83E7 0B13h must be written to KICK0 as part of the process to unlock the CPU write access to the PINMMRn registers.

3.5.4 KICK_REG1: Kicker Register 1

This register forms the second part of the unlock sequence for being able to update the I/O multiplexing control registers (PINMMRnn).

Figure 3-6. KICK_REG1: Kicker Register 1 (Offset = 3Ch)



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 3-5. Kicker Register 1 Field Descriptions

Bit	Field	Value	Description
31-0	KICK1	0	Kicker 1 Register. The value 95A4 F1E0h must be written to the KICK1 as part of the process to unlock the CPU write access to the PINMMRn registers.

3.5.5 ERR_RAW_STATUS_REG: Error Raw Status / Set Register

This register shows the statuses of the error conditions (before enabling) and allows setting the status.

Figure 3-7. ERR_RAW_STATUS_REG: Error Raw Status / Set Register (Offset = E0h)

31	8
Reserved	
R-0	

7	2	1	0
Reserved		ADDR_ERR	PROT_ERR
R-0		R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 3-6. Error Raw Status / Set Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Read returns zeros, writes have no effect.
1	ADDR_ERR		Addressing Error Status. An Addressing Error occurs when an unimplemented location inside the IOMM register frame is accessed.
		0	Read: Addressing Error has not occurred. Write: Writing 0 has no effect.
		1	Read: Addressing Error has been detected. Write: Addressing Error status is set.
0	PROT_ERR		Protection Error Status. A Protection Error occurs when any control register inside the IOMM is written in the CPU's user mode of operation.
		0	Read: Protection Error has not occurred. Write: Writing 0 has no effect.
		1	Read: Protection Error has been detected. Write: Protection Error status is set.

3.5.6 ERR_ENABLED_STATUS_REG: Error Enabled Status / Clear Register

This register shows the error signal enabled status and allows clearing of the error status.

Figure 3-8. ERR_ENABLED_STATUS_REG: Error Enabled Status / Clear Register (Offset = E4h)

31	8
Reserved	
R-0	

7	2	1	0
Reserved		ENABLED_ADDR_ERR	ENABLED_PROT_ERR
R-0		R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 3-7. Error Signaling Enabled Status / Clear Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Read returns zeros, writes have no effect.
1	ENABLED_ADDR_ERR		Addressing Error Signaling Enable Status and Status Clear
		0	Read: Addressing Error Signaling is disabled.
			Write: Writing 0 has no effect.
		1	Read: Addressing Error Signaling is enabled.
			Write: Addressing Error status is cleared.
0	ENABLED_PROT_ERR		Protection Error Signaling Enable Status and Status Clear
		0	Read: Protection Error Signaling is disabled.
			Write: Writing 0 has no effect.
		1	Read: Protection Error Signaling is enabled.
			Write: Protection Error status is cleared.

3.5.7 ERR_ENABLE_REG: Error Signaling Enable Register

This register shows the interrupt enable status and allows enabling of the interrupts.

Figure 3-9. ERR_ENABLE_REG: Error Signaling Enable Register (Offset = E8h)

31	8
Reserved	
R-0	

7	2	1	0
Reserved		ADDR_ERR_EN	PROT_ERR_EN
R-0		R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 3-8. Error Enable Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Read returns zeros, writes have no effect.
1	ADDR_ERR_EN		Addressing Error Signaling Enable
		0	Read: Addressing Error Signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Addressing Error Signaling is enabled. Write: Addressing Error Signaling is enabled.
0	PROT_ERR_EN		Protection Error Signaling Enable
		0	Read: Protection Error Signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Protection Error Signaling is enabled. Write: Protection Error Signaling is enabled.

3.5.8 ERR_ENABLE_CLR_REG: Error Signaling Enable Clear Register

This register shows the error signaling enable status and allows disabling of the error signaling.

Figure 3-10. ERR_ENABLE_CLR_REG: Error Signaling Enable Clear Register (Offset = ECh)

31	8
Reserved	
R-0	

7	2	1	0
Reserved		ADDR_ERR_EN_CLR	PROT_ERR_EN_CLR
R-0		R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 3-9. Interrupt Enable Clear Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Read returns zeros, writes have no effect.
1	ADDR_ERR_EN_CLR		Addressing Error Signaling Enable Clear
		0	Read: Addressing Error signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Addressing Error signaling is enabled. Write: Addressing Error signaling is disabled.
0	PROT_ERR_EN_CLR		Protection Error Signaling Enable Clear
		0	Read: Protection Error signaling is disabled. Write: Writing 0 has no effect.
		1	Read: Protection Error signaling is enabled. Write: Protection Error signaling is disabled.

3.5.9 FAULT_ADDRESS_REG: Fault Address Register

This register holds the address of the first fault transfer.

Figure 3-11. FAULT_ADDRESS_REG: Fault Address Register (Offset = F4h)

31	16
Reserved	
R-0	

15	9	8	0
Reserved		FAULT_ADDR	
R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 3-10. Fault Address Register Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Read returns zeros, writes have no effect.
8-0	FAULT_ADDR	0	Fault Address. This field captures the fault address in case of an address error or a protection error condition.

3.5.10 FAULT_STATUS_REG: Fault Status Register

This register holds the status and attributes of the first fault transfer.

Figure 3-12. FAULT_STATUS_REG: Fault Status Register (Offset = F8h)

31	28	27	24	23	16
Reserved		FAULT_ID		FAULT_MSTID	
R-0		R-0		R-0	

15	13	12	9	8	7	6	5	0
Reserved		FAULT_PRIVID		Rsvd	FAULT_NS	Rsvd	FAULT_TYPE	
R-0		R-0		R-0	R-0	R-0	R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 3-11. Fault Status Register Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return zeros, writes have no effect.
27-24	FAULT_ID	0-Fh	Faulting Transaction ID
23-16	FAULT_MSTID	0-FFh	ID of Master that initiated the faulting transaction
15-13	Reserved	0	Reads return zeros, writes have no effect.
12-9	FAULT_PRIVID	0-Fh	Faulting Privilege ID
8	Reserved	0	Reads return zeros, writes have no effect.
7	FAULT_NS	0	Fault: Non-secure access is detected
6	Reserved	0	Reads return zeros, writes have no effect.
5-0	FAULT_TYPE		Type of fault detected.
		0	No fault
		1h	User execute fault
		2h	User write fault
		4h	User read fault
		8h	Supervisor execute fault
		10h	Supervisor write fault
		20h	Supervisor read fault
		others	Reserved

3.5.11 FAULT_CLEAR_REG: Fault Clear Register

This register allows the application to clear the current fault so that another can be captured when 1 is written to this register.

Figure 3-13. FAULT_CLEAR_REG: Fault Clear Register (Offset = FCh)

31	16
Reserved	
R-0	

15	1	0
Reserved		FAULT_CLEAR
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 3-12. FAULT_CLEAR_REG: Fault Clear Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zeros, writes have no effect.
0	FAULT_CLEAR		Fault Clear
		0	Read: Current value of the FAULT_CLEAR bit is 0. Write: Writing 0 has no effect.
		1	Read: Current value of the FAULT_CLEAR bit is 1. Write: Writing a 1 clears the current fault.

3.5.12 PINMMRn: Pin Multiplexing Control Registers

These registers control the multiplexing of the functionality available on each pad. There are 31 registers, PINMMR0 through PINMMR30. PINMMR7 and PINMMR10 are implemented and accessible on this device, but are not mapped and connected to anything functionally; the default value of PINMMR7 = 0000 0001h and the default value of PINMMR10 = 0001 0101h. PINMMR11 through PINMMR30 are not implemented on this device. Each 8-bit field of a PINMMR register controls the functionality of a single ball/pin. The mapping between the PINMMRn control registers and the functionality selected on a given terminal is defined in Table 3-14.

Figure 3-14. PINMMRn: Pin Multiplexing Control Registers (Offset = 110h-138h)

31	24	23	16
PINMMRn[31–24]		PINMMRxn23–16]	
R/WP-1		R/WP-1	

15	8	7	0
PINMMRn[15–8]		PINMMRn[7–0]	
R/WP-1		R/WP-1	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 3-13. Pin Multiplexing Control Registers Field Descriptions

Bit	Field	Value	Description
31-24	PINMMRn[31–24]	1h	Each of these byte-fields controls the functionality on a given ball/pin. Refer to Table 3-14 for a list of multiplexed signals sorted by the control registers.
23-16	PINMMRn[23–16]	1h	
15-8	PINMMRn[15–8]	1h	
7-0	PINMMRn[7–0]	1h	

Table 3-14. Multiplexing and Control

Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit
GIOA[0]	PINMMR0[8]	SPI3nCS[3]	PINMMR0[9]	-	-
GIOA[1]	PINMMR1[0]	SPI3nCS[2]	PINMMR1[1]	-	-
GIOA[2]	PINMMR1[8]	SPI3nCS[1]	PINMMR1[9]	-	-
GIOA[3]	PINMMR1[16]	SPI2nCS[3]	PINMMR1[17]	-	-
GIOA[4]	PINMMR1[24]	SPI2nCS[2]	PINMMR1[25]	-	-
GIOA[5]	PINMMR2[0]	EXTCLKIN	PINMMR2[1]	-	-
GIOA[6]	PINMMR2[8]	SPI2nCS[1]	PINMMR2[9]	N2HET[31]	PINMMR2[10]
GIOA[7]	PINMMR2[16]	N2HET[29]	PINMMR2[17]	-	-
MIBSPI1nCS[2]	PINMMR3[0]	N2HET[20]	PINMMR3[1]	N2HET[19]	PINMMR3[2]
SPI3CLK	PINMMR3[16]	EQEPA	PINMMR3[17]	-	-
SPI3nENA	PINMMR3[24]	EQEPB	PINMMR3[25]	-	-
SPI3nCS[0]	PINMMR4[0]	EQEPI	PINMMR4[1]	-	-
MIBSPI1nCS[3]	PINMMR4[8]	N2HET[26]	PINMMR4[9]	-	-
ADEVT	PINMMR4[16]	N2HET[28]	PINMMR4[17]	-	-
MIBSPI1nENA	PINMMR5[8]	N2HET[23]	PINMMR5[9]	NHET[30]	PINMMR5[10]
MIBSPI1nCS[1]	PINMMR6[8]	EQEPS	PINMMR6[9]	N2HET[17]	PINMMR6[10]

F021 Flash Module Controller

The Flash electrically-erasable programmable read-only memory module is a type of nonvolatile memory that has fast read access times and is able to be reprogrammed in the field or in the application. This chapter describes the F021 Flash module controller (FMC).

Topic **Page**

4.1 Overview

The F021 Flash is used to provide non-volatile memory for instruction execution or data storage. The Flash can be electrically programmed and erased many times to ease code development.

Refer to the following documents for support on how to initialize and use the on-chip Flash and its API:

- *Initialization of the TMS570LS043x, TMS570LS033x and RM42L432 Hercules ARM Cortex-R4 Microcontrollers Application Report* (SPNA163)
- *F021 (Texas Instruments 65nm Flash) Flash API Reference Guide* (SPNU501)

4.1.1 Features

- Read, program and erase with a single 3.3 V supply voltage
- Supports error detection and correction
  - Single Error Correction and Double Error Detection (SECDED)
  - Error Correction Code (ECC) is evaluated in the CPU for the main Flash bank arrays and in the Flash Wrapper for the EEPROM emulation Flash banks
  - Address bits included in ECC calculation
- Provides different read modes to optimize performance and verify the integrity of Flash contents
- Provides built-in power mode control logic
- Integrated program/erase state machine
  - Simplifies software algorithms
  - Supports simultaneous read access on a bank while performing a write or erase operation on any one of the remaining banks
  - Suspend command allows read access to a sector being programmed/erased
  - Fast erase and program times (for details, see the device-specific data sheet)

For the actual size of the Flash memory for the device, see the device-specific data sheet.

4.1.2 Definition of Terms

Terms used in this document have the following meaning:

- ATCM: Port A tightly coupled memory
- BAGP (Bank Active Grace Period): Time (in HCLK cycles) from the most recent Flash access of a particular bank until that bank enters fallback power mode. This reduces power consumption by the Flash. However, it can also increase access time.
- bw: Normal data space bank data width of a Flash bank. The bw is 128 bits (144 bits including the error correction bits).
- bwe: EEPROM emulation bank is 64-bits wide (72 bits including the error correction bits).
- Charge pump: Voltage generators and associated control (logic, oscillator, and bandgap, for example).
- CSM: Program/erase command state machine
- Fallback power mode: The power mode (active, standby or sleep, depending on which mode is selected) into which a bank or the charge pump falls back each time the active grace period expires.
- Flash bank: A group of Flash sectors that share input/output buffers, data paths, sense amplifiers, and control logic.
- FEE: Flash EEPROM Emulation. Features on the FMC to support using a Flash type memory in place of an EEPROM Flash memory. EEPROM is erasable by the word while this Flash memory is only erasable by the sector. The FEE bank is accessible only through Bus 2 in a special address range and always resides in bank 7.
- Flash module: Flash banks, charge pump, and Flash wrapper.
- Flash wrapper: Power and mode control logic, data path, wait logic, and write/erase state machines.
- FMC: Flash Module Controller.
- Command: A sequence of coded instructions to Flash module to execute a certain task.

- FSM (Flash State Machine) - State machine that parses and decodes FSM commands. It executes embedded algorithms and generates control signals to both Flash bank and charge pump during the actual program/erase operation.
- OTP (one-time programmable): A program-only-once Flash sector (cannot be erased)
- PAGP (Pump Active Grace Period): Time (in HCLK cycles) from when the last of the banks have entered fallback power mode until the pump enters a fallback power mode. This can reduce power consumption by the Flash; however, it can also increase access time.
- Pipeline mode: The mode in which Flash is read 128 bits (+ 16 bit ECC) at a time, providing higher throughput.
- Sector: A contiguous region of Flash memory that must be erased simultaneously.
- Wide_Word: The width of the data output from the Flash bank. This is 144-bits wide for main Flash banks and 72 bits wide for the FEE bank.
- Standard read mode: The mode assumed when the pipeline mode is disabled. Physically, 128 (+ 16 bit ECC) is read at a time. However, only 32 bits of data is used while the other bits of data are discarded.
- Read Margin 1 mode: More stringent read mode designed for early detection of marginally erased bits.
- Read Margin 0 mode: More stringent read mode designed for early detection of marginally programmed bits.

4.1.3 F021 Flash Tools

Texas Instruments provides the following tools for F021 Flash:

- nowECC Generation Tool - to generate the Flash ECC from the Flash data.
- nowFLASH Programming Tool - to erase/program/verify the device Flash content through JTAG.
- Code Composer Studio - the development environment with integrated Flash programming capabilities.
- F021 Flash API Library - a set of software peripheral functions to program/erase the Flash module. Refer to *F021 Flash API Reference Guide* (SPNU501) for more information.

4.2 Default Flash Configuration

At power up, the Flash module state exhibits the following properties:

- Wait states are set to 1 data wait state and 0 address wait states
- Pipeline mode is disabled
- The Flash content is protected from modification
- Power modes are set to *Active* (no power savings)
- The boot code must initialize the wait states (including data wait states and address wait states) and the desired pipeline mode by initializing the FRDCNTL register to achieve the optimum system performance. This needs to be done before switching to the final device operating frequency. Refer to *Initialization of the TMS570LS043x, TMS570LS033x and RM42L432 Hercules ARM Cortex-R4 Microcontrollers Application Report* (SPNA163) for more information.

4.3 SECDED

The Flash memory can be protected by Single Error Correction Double Error Detection (SECDED). The main program memory is protected by the SECDED circuit inside of the Cortex-R4 CPU. All OTP and the FEE memory (bank 7) is protected by SECDED logic in the Flash wrapper.

4.3.1 SECDED Initialization

Flash error detection and correction is not enabled at reset. To enable SECDED, error correction detection must be enabled in the Flash wrapper, the CPU event bus must be enabled and SECDED must be enabled within the CPU. Refer to *Initialization of the TMS570LS043x, TMS570LS033x and RM42L432 Hercules ARM Cortex-R4 Microcontrollers Application Report* (SPNA163) for information on these steps.

The ECC values for all of the ATCM program-memory space (Flash banks 0 through 6) must be programmed into the Flash before SECDED is enabled. This can be done by generating the correct values of the ECC with an external tool such as nowECC or may be generated by the programming tool. The Cortex-R4 CPU may generate speculative fetches to any location within the ATCM memory space. A speculative fetch to a location with invalid ECC, which is subsequently not used, will not create an abort, but will set the ESM flags for a correctable or uncorrectable error. An uncorrectable error will unconditionally cause the nERROR pin to toggle low. Therefore care must be taken to generate the correct ECC for the entire ATCM space including the holes between sections and any unused or blank Flash areas.

The Cortex-R4 CPU does not generate speculative fetches into the address space of bank 7, the EEPROM Emulation Flash. It is only necessary to initialize the ECC values of the locations which will be intentionally read by the CPU or other bus masters.

NOTE: The memory region from 0x0002_0000 to 0x0003_FFFF is not inherently protected against creating nERROR pin toggles from speculative fetches or from run away code. An MPU region must be used to limit accesses to only the first 128KB of ATCM space (0x0000_0000 to 0x0001_FFFF). Also, Flash API version 02.01.01 or later should be used to program or erase the Flash of this device.

4.3.2 ECC Encoding

Nineteen address lines are also included in the ECC calculation. A failure of a single address line inside of the bank will be treated as an uncorrectable error. The ECC encoding is show in Table 4-1. Bits 31:0 come from the word at the address ending in 0x0 or 0x8, Bits 63:31 come from the word at the address ending in 0x4 or 0xC.

Table 4-1. ECC Encoding for BE32 Devices

		82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
		Participating Address Bits																		
ADDR_MSW_LSW	**ECC Bit**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**
0007F_00FFFF00_FF0000FF	7													x	x	x	x	x	x	x
7FF80_FF0000FF_FF0000FF	6	x	x	x	x	x	x	x	x	x	x	x	x							
07F80_00FF00FF_00FF00FF	5					x	x	x	x	x	x	x	x							
19F83_FCC0FCC0_FCC0FCC0	4			x	x			x	x	x	x	x	x						x	x
6A78D_E338E338_E338E338	3	x	x		x		x			x	x	x	x				x	x		x
2A9B5_99A699A6_99A699A6	2		x		x		x		x			x	x		x	x		x		x
0BAD1_57155715_57155715	1				x		x	x	x		x		x	x		x				x
554EA_D1B4D1B4_2E4B2E4B	0	x		x		x		x		x			x	x	x		x		x	

Participating Data Bits																																				
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**	**31**	**30**	**29**	**28**	**27**
								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	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			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	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	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

Participating Data Bits																											**Parity (1)**	**Check Bits (2)**
26	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**		
x	x	x																	x	x	x	x	x	x	x	x	Even	ECC[7]
x	x	x																	x	x	x	x	x	x	x	x	Even	ECC[6]
			x	x	x	x	x	x	x	x									x	x	x	x	x	x	x	x	Even	ECC[5]
x			x	x							x	x	x	x	x	x			x	x							Even	ECC[4]
	x	x			x	x	x				x	x	x				x	x			x	x	x				Odd	ECC[3]
		x	x		x			x	x		x			x	x			x	x		x			x	x		Odd	ECC[2]
x	x	x				x		x		x		x		x		x	x	x				x		x		x	Even	ECC[1]
x	x			x			x		x	x			x		x	x	x			x			x		x	x	Even	ECC[0]

(1) For Odd parity, XOR a 1 to the row's XOR result. For even Parity, use the row's XOR result directly.
(2) Each ECC[x] bit represents the XOR of all the address and data bits marked with x in the same row.

4.3.3 Syndrome Table: Decode to Bit in Error

The syndrome is a 8 bit value which decodes to the bit in error. The bit in error can be a bit among the 64 data bits, the 19 address bits or a bit among the 8 ECC check bits. A syndrome value of 00000000 indicates there is no error. Any other syndrome combinations not shown in the table are un-correctable multi-bit error. Errors of three of more bits may escape detection. The syndrome decoding is show in Table 4-2.

Table 4-2. Syndrome Table, Decode to Bit in Error

	Address Bit Error Position																		
	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03
	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	1	1	1	0	0	0	0	0	0	0
	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1
	1	1	0	1	0	1	0	0	1	1	1	1	0	0	0	1	1	0	1
	0	1	0	1	0	1	0	1	0	0	1	1	0	1	1	0	1	0	1
	0	0	0	1	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1
	1	0	1	0	1	0	1	0	1	0	0	1	1	1	0	1	0	1	0

Data Bit Error Position																																								
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	31	30	29	28	27
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	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	1	1	1	1	1	1	1	1	1	1	1	1	1	
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	
1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	1	1	1	1	1	
1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1	0	0
1	0	0	1	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	1	1
0	1	0	1	0	1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	0
1	1	0	1	0	0	0	1	1	0	1	1	0	1	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	1	0	1

Data Bit Error Position																															ECC Error Bit								
26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	07	06	05	04	03	02	01	00	
1	1	1	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	0	0	0	0	0	0	0	Bit[7]
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	Bit[6]
0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	Bit[5]
1	0	0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	Bit[4]
0	1	1	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	Bit[3]
0	0	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0	0	1	1	0	1	0	0	1	1	0	0	0	0	0	0	1	0	0	Bit[2]
1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	Bit1]
1	1	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	1	Bit[0]

4.3.4 Syndrome Table: An Alternate Method

Table 4-3. Alternate Syndrome Table

Syndrome lsb: 3:0	Syndrome msb 7:4															
	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	good	E04	E05	D	E06	D	D	D38	E07	D	D	D54	D	M	M	D
x1	E00	D	D	D22	D	A19	A17	D	D	A04	M	D	M	D	D	D06
x2	E01	D	D	M	D	D58	D32	D	D	D42	D48	D	M	D	D	M
x3	D	D10	D16	D	M	D	D	A15	A09	D	D	M	D	D26	D00	D
x4	E02	D	D	D23	D	D59	D33	D	D	D43	D49	D	M	D	D	D07
x5	D	D11	D17	D	M	D	D	D39	A08	D	D	D55	D	D27	D01	D
x6	D	D12	D18	D	M	D	D	A14	A07	D	D	M	D	D28	D02	D
x7	M	D	D	M	D	D60	D34	D	D	D44	D50	D	M	D	D	M
x8	E03	D	D	M	D	D61	D35	D	D	D45	D51	D	M	D	D	M
x9	D	D13	D19	D	A21	D	D	A13	A06	D	D	M	D	D29	D03	D
xA	D	D14	D20	D	D57	D	D	A12	D41	D	D	M	D	D30	D04	D
xB	D09	D	D	M	D	D62	D36	D	D	D46	D52	D	D25	D	D	M
xC	D	D15	D21	D	A20	D	D	A11	A05	D	D	M	D	D31	D05	D
xD	M	D	D	M	D	D63	D37	D	D	D47	D53	D	M	D	D	M
xE	D08	D	D	M	D	A18	A16	D	D	A03	M	D	D24	D	D	M
xF	D	M	M	D	D56	D	D	A10	D40	D	D	M	D	M	M	D

- E0x - Single-bit ECC error, correctable
- Dxx - Single-bit data error, correctable
- Axx - Single-bit address error, uncorrectable
- D - Double-bit error, uncorrectable
- M - Multi-bit errors, uncorrectable

4.4 Memory Map

The Flash module contains the program memory, which is mapped starting at location 0, and one Customer OTP sector and one TI OTP sector per bank. The Customer OTP sectors may be programmed by the customer, but cannot be erased. They are typically blank in new parts. The TI OTP sectors are used to contain manufacturing information. They may be read by the customer but can not be programmed or erased. The TI OTP sectors contain settings used by the Flash API to setup the Flash state machine for erase and program operations.

All of these OTP regions are memory-mapped to facilitate ease of access by the CPU. They are memory mapped to an offset starting at F000 0000h in the CPUs memory map.

The RWAIT value is used to define the number of wait states for the program-memory Flash. The EWAIT value is used to define the number of wait states for the data Flash in bank 7. Bank 7 starting at offset F020 0000h is dedicated for data storages such as EEPROM Emulation.

For the Flash Bank/Sectoring information, see the device-specific data sheet.

4.4.1 Location of Flash ECC Bits

The ECC bits are packed in their memory space as shown in Figure 4-1 and Figure 4-2.

The ECC bytes for bank 0 must be read as bytes or halfwords. Reading a single ECC byte from bank 0 with ECC enabled will actually cause 144 bits to be read from the Flash, and the ECC bits will be corrected if necessary. Any errors in either of the two double-words accessed will be recorded in the FEDACSTATUS register.

The ECC bytes for bank 7 must be read as bytes only. Reading a single ECC byte from bank 7 with ECC enabled will actually cause 72 bits to be read from the Flash, and the ECC bits will be corrected if necessary. Any errors in the double-word accessed will be recorded in the EE_STATUS register.

Figure 4-1. ECC Organization for Program Flash (144-Bits Wide)

Big Endian

	64 – bit data word
0x00000028	64 – bit data word 5
0x00000020	64 – bit data word 4
0x00000018	64 – bit data word 3
0x00000010	64 – bit data word 2
0x00000008	64 – bit data word 1
0x00000000	64 – bit data word 0

16-bit Read

0xF0400004	ECC4	ECC5
0xF0400002	ECC2	ECC3
0xF0400000	ECC0	ECC1

8-bit Read

0xF0400005	ECC5
0xF0400004	ECC4
0xF0400003	ECC3
0xF0400002	ECC2
0xF0400001	ECC1
0xF0400000	ECC0

Figure 4-2. ECC Organization for Bank 7 (72-Bits Wide)

Big Endian

	64 – bit data word
0xF0200028	64 – bit data word 5
0xF0200020	64 – bit data word 4
0xF0200018	64 – bit data word 3
0xF0200010	64 – bit data word 2
0xF0200008	64 – bit data word 1
0xF0200000	64 – bit data word 0

8-bit Read

0xF0100005	ECC5
0xF0100004	ECC4
0xF0100003	ECC3
0xF0100002	ECC2
0xF0100001	ECC1
0xF0100000	ECC0

4.4.2 OTP Memory

4.4.2.1 Flash Bank and Sector Sizes

Flash Bank/Sectoring information can be determined from the device-specific datasheet or can be computed by reading locations in the TI OTP and FMC registers.

The number of banks, which banks are available, and the number of sectors for bank 0 can be read from TI OTP location F008 0158h as shown in Figure 4-3 and described in Table 4-4.

The bank sector information is repeated once for each bank in the device. The number of sectors is unique for each bank. The number of banks and which banks are implemented is repeated in each location. Use the TI OTP information for bank 0 to determine which banks are in the device, and then read the number of sectors for each bank using the TI OTP locations shown in Table 4-5.

Bank 0 will start at address 0000 0000h. Bank 7 will start at address F020 0000h. Refer to Table 4-5 to find the starting address of other banks. Refer to Table 4-4 to find the sector sizes in each bank.

Figure 4-3. TI OTP Bank 0 Sector Information

31							24	23							16
Reserved								BX_NUM_Sectors							
R								R							
15	14	13	12	11	10	9	8	7							0
B7	B6	B5	B4	B3	B2	B1	B0	NUM_Banks							
R-1	R-0	R-0	R-0	R-0	R-0	R-1	R-1	R							

LEGEND: R = Read only

Table 4-4. TI OTP Bank 0 Sector Information Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved. All bits will be read as 0.
23-16	BX_NUM_Sectors	1-32	Number of sectors in this bank.
15	B7	1	1 = Bank 7 is present
14	B6	0	0 = Bank 6 is not present
13	B5	0	0 = Bank 5 is not present
12	B4	0	0 = Bank 4 is not present
11	B3	0	0 = Bank 3 is not present
10	B2	0	0 = Bank 2 is not present
9	B1	1	1 = Bank 1 is present
8	B0	1	1 = Bank 0 is present
7-0	NUM_Banks	2 or 3	Number of banks on this part.

Table 4-5. TI OTP Sector Information Address

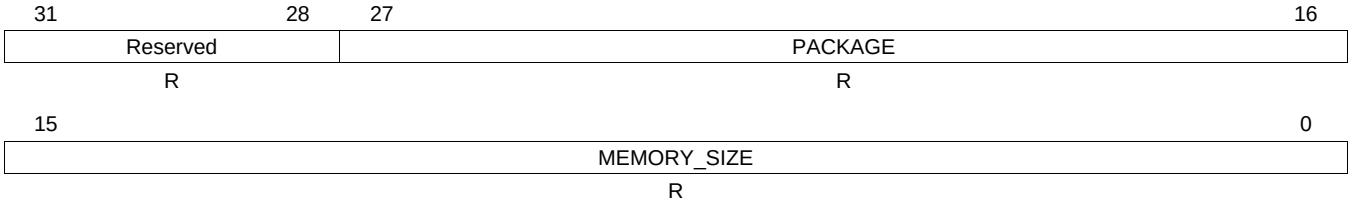
Bank	TI OTP Address
0	F008 0158h
1	F008 2158h
2	F008 4158h
3	F008 6158h
4	F008 8158h
5	F008 A158h
6	F008 C158h
7	F008 E158h

4.4.2.2 Package and Memory Size

Package and memory size information can be determined from the device-specific datasheet, or can be computed by reading locations in the TI OTP Bank 0 registers.

The package and memory size can be read from TI OTP location F008 015Ch as shown in Figure 4-4 and described in Table 4-6.

Figure 4-4. TI OTP Bank 0 Package and Memory Size Information (F008 015Ch)



LEGEND: R = Read only

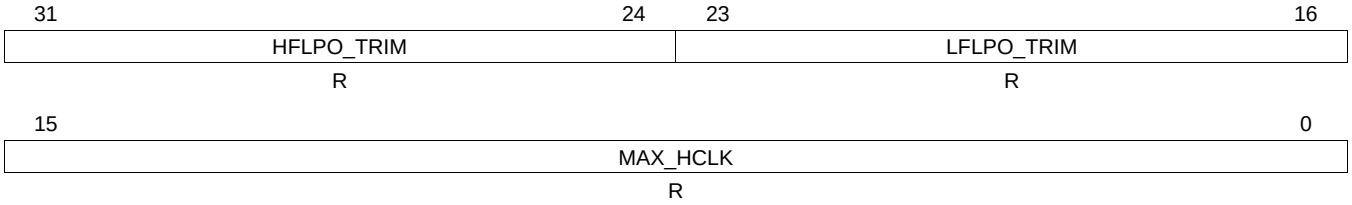
Table 4-6. TI OTP Bank 0 Package and Memory Size Information Field Descriptions

Bit	Field	Description
31-28	Reserved	Reserved
27-16	PACKAGE	Count of pins in the package
15-0	MEMORY_SIZE	Flash memory size in Kbytes

4.4.2.3 LPO Trim and Max HCLK

The HF LPO trim solution, LF LPO trim solution and maximum HCLK frequency can be read from TI OTP location F008 01B4h as shown in Figure 4-5 and described in Table 4-7.

Figure 4-5. TI OTP Bank 0 LPO Trim and Max HCLK Information (F008 01B4h)



LEGEND: R = Read only

Table 4-7. TI OTP Bank 0 LPO Trim and Max HCLK Information Field Descriptions

Bit	Field	Description
31-24	HFLPO_TRIM	HF LPO Trim Solution
23-16	LFLPO_TRIM	LF LPO Trim Solution
15-0	MAX_HCLK	Maximum HCLK Speed

4.4.2.4 Part Number Symbolization

Device part number symbolization information can be determined from the device-specific datasheet or can be computed by reading locations in the TI OTP bank 0 registers.

The device part number symbolization can be read from TI OTP bank 0 location F008 01E0h through F008 01FFh, as shown in Figure 4-6.

Figure 4-6. TI OTP Bank 0 Symbolization Information (F008 01E0h-F008 01FFh)

0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x54	0x4D	0x53	0x35	0x37	0x30	0x4C	0x53	0x33	0x31	0x33	0x37	0x43	0x50	0x47	0x45
R															

0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
0x51	0x51	0x31	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
R															

LEGEND: R = Read only

4.4.2.5 Deliberate ECC Errors for FMC ECC Checking

Deliberate single-bit and double-bit errors have been placed in the OTP for checking the FMC ECC functionality. Any portion of the 64 bits in TI OTP bank 0 location F008 03F0h through F008 03F7h as shown in Figure 4-7 will generate a single-bit error. Any portion of the 64 bits in TI OTP bank 0 location F008 03F8h through F008 03FFh as shown in Figure 4-7 will generate a double-bit error.

Figure 4-7. TI OTP Bank 0 Deliberate ECC Error Information (F008 03F0h-F008 03FFh)

0x00	0x04	0x08	0x0C
0x12345678	0x9ABCDEF1	0x12345678	0x9ABCDEF3
R	R	R	R

LEGEND: R = Read only, ECC is calculated for the value 0x123456789ABCDEF0

4.5 Power On, Power Off Considerations

4.5.1 Error Checking at Power On

As the device is coming out of the device reset sequence the Flash wrapper reads two configuration words from the TI OTP section of bank 0. During these reads ECC is enabled. Single-bit errors are corrected and uncorrectable errors will generate an ESM group 3 channel 7 error event. The $\overline{\text{ERROR}}$ pin

4.5.2 Flash Integrity at Power Off

If power is lost during a programming or erase operation, a power-on reset must be asserted before the core supply voltage drops below specification. The $\overline{\text{PORRST}}$ pin has a glitch filter that means that the $\overline{\text{PORRST}}$ pin must be asserted low $t_{f(nPORRST)}$ (2 µs) before the core supply drops below Vcc_{MIN} (1.14V). If this requirement is met, then the bits being programmed when $\overline{\text{PORRST}}$ goes low are indeterminate; however, the other bits in the Flash are not disturbed. Likewise, if this requirement is met, and $\overline{\text{PORRST}}$ is asserted while erasing, the sector or sectors being erased will have indeterminate bits; however, the other sectors in the same bank and the other banks will not be disturbed.

4.6 Emulation and SIL3 Diagnostic Modes

4.6.1 System Emulation

During emulation when the SUSPEND signal is high, the data read from memory is still passed to SECDED for correction if ECC_ENABLE is active. If a correctable error is detected, then it is corrected but an error event is not generated and the error occurrence counter is not incremented if in profiling mode. If a double error is detected, then the raw data is returned without generating a double-error signal.

The SUSPEND signal can be disable by using the SUSP_IGNR bit in the Flash error detection and correction control register 1 (FEDACCTRL1). The SUSPEND signal should not be confused with the suspend_now operation for the FSM.

4.6.2 Diagnostic Mode

The Flash wrapper can be put in diagnostic mode to verify various logic. There are multiple diagnostic modes supported by the wrapper. A specific diagnostic mode is selected via the DIAG_MODE control bits in the diagnostic control register (FDIAGCTRL), as listed in Table 4-8.

The diagnostic mode is only enabled by a 4-bit key stored in the DIAG_EN_KEY bits in FDIAGCTRL register. Only DIAG_EN_KEY = 5h enables any diagnostic mode and all diagnostic modes use the DIAG_TRIG bit in FDIAGCTRL register to initiate the action.

All tests run from any pipeline mode. Some of the diagnostic modes can corrupt the Flash data access, and generate errors as part of the test. Running in non-pipeline may minimize some of these conditions.

For all modes it is best to follow this sequence:

1. Write 5h to the DIAG_EN_KEY bits and set the desired DIAG_MODE control bits. This blocks many UERR sources.
2. Set any data registers needed for this mode.
3. Write 1 to the DIAG_TRIG bit to initiate the action and allow UERRs to happen for one cycle.
4. Write Ah to the DIAG_EN_KEY bits to disable the diagnostic modes.

When the CONF_TYPE is 5 then the ECC logic is in the CPU and most of diagnostic modes are removed. Some because the logic they test no longer exists and the others because the path is validated via the ECC path to the CPU.

Table 4-8. DIAG_MODE Encoding

Mode	DIAG_MODE Bits			Description
0	0	0	0	Diagnostic mode is disabled. Same as DIAG_EN_KEY not equal to 5h.
1	0	0	1	ECC Data Correction test mode
2	0	1	0	ECC Syndrome Reporting test mode
3	0	1	1	ECC Malfunction test mode 1 (same data)
4	1	0	0	ECC Malfunction test mode 2 (inverted data)
5	1	0	1	Address Tag Register test mode
6	1	1	0	Reserved
7	1	1	1	ECC Data Correction Diagnostic test mode

4.6.2.1 ECC Data Correction Test Mode: DIAG_MODE = 1

This diagnostic mode can be enabled while ECC logic is also enabled for normal bank read. The Flash wrapper will arbitrate the usage of the ECC logic if a conflict occurs between a normal bank read and diagnostic checking.

When in diagnostic data correction mode, FEMU_xxxx registers contain the 64-bit EEPROM emulation data register, the 19-bit emulation address register and the 8-bit emulation check-bit register. These values are used to enter diagnostic data to exercise the SECDED logic. The user can apply a value with an error in any bit location. When the DIAG_TRIG is set, the SECDED calculation is done and the corrected values are saved back into the same FEMU_xxxx registers. The error position register is also updated to indicate the bit position in error. Either ERR_ONE_FLG or ERR_ZERO_FLG bit is set when a correctable error is detected. The D_COR_ERR bit will also be set in FEDACSTATUS register. For uncorrectable error, the error status bit ERR_PRF_FLG is set as well as the D_UNC_ERR bit in the same register. Status bits should be cleared by the user before applying a new diagnostic data.

It takes multiple CPU transactions to preload the registers with diagnostic values. During this time, the result of the diagnostic logic such as comparator can change. User should apply a trigger by setting DIAG_TRIG bit to 1 as a qualifier after all registers are loaded with intended values. The DIAG_TRIG serves to validate the diagnostic result. Only when DIAG_TRIG is high and a failing result in the diagnostic logic will update the corresponding status flag and the position register.

4.6.2.2 ECC Syndrome Reporting Test Mode: DIAG_MODE = 2

When in diagnostic syndrome reporting mode, the resulting syndrome calculated by SECDED is captured into the ECC check-bit register FEMU_ECC. The syndrome can be read by the user and compare with a known syndrome value. Diagnostic data in FEMU_DxSW and FEMU_ADDR is not corrected and the error position register is not updated. The FEDACSTATUS register error bits are not updated during this mode.

For devices with ECC_IN_CPU (CONF_TYPE = 5), the resulting FEMU_ECC value represents the 32-bit byte swapped values. Here, bytes 7654_3210 are rearranged to 4567_0123. For instance, if the syndrome shows an error in data bit 33, it would really be an error in EMU_DMW bit 57. You can also XOR the data bit position with “011000”. (21h XOR 18h => 39h)

NOTE: The user should pre-load the registers with the test values with DIAG_TRIG = 0. After all test values are written, the DIAG_TRIG should then be set high to validate the diagnostic result.

4.6.2.3 ECC Malfunction Test Mode 1: DIAG_MODE = 3

There are three inputs to the malfunction detection logic: the resulting syndrome, the original uncorrected data, and the final corrected data.

In normal function, the malfunction detection logic will detect an error if the syndrome is 0 and if the data before correction and the data after the correction is not equal or if the syndrome is not 0 and if the data before correction and data after the correction is equal to each other. During diagnostic mode 3 or 4, user supplied values are sent to the malfunction logic. No functional checking is done by the ecc_malfunction logic while the mode is 3 or 4.

Diagnostic mode 3 is also known as “same data” mode. A diagnostic value can be stored in the ECC checkbit register. The value stored in the 64-bit Raw data register will be supplied to the two inputs of the malfunction comparator logic. If a non-zero value is stored in the Raw ECC checkbit register (FRAW_ECC), then the malfunction logic should detect it as an error. The DIAG_TRIG is set to initiate this mode.

4.6.2.4 ECC Malfunction Test Mode 2: DIAG_MODE = 4

This mode is also known as “inverted data” mode. A diagnostic value can be stored in the ECC checkbit register. A value stored in the 64-bit Raw data register and its bit-wise inverted counterpart will be supplied to the two inputs of the malfunction comparator logic. If a zero value is stored in the Raw ECC checkbit register (FRAW_ECC), then the malfunction logic should detect it as an error.

Set the DIAG_ECC_SEL bits before entering mode 4 or enter mode 4 from a non-mode 4 and set the DIAG_ECC_SEL bits at the same time.

4.6.2.5 Address Tag Register Test Mode: DIAG_MODE = 5

NOTE: The test code for Diag mode 5 needs to be executed from RAM or when executed from Flash, the Flash Address Wait State (ASWSTEN) bit in the FRDCNTL register should be set to 1. This is due to conflicting accesses by the CPU and Flash wrapper during the test execution.

There are four sets of address tag registers. Each set consists of a primary and a duplicate address tag registers. Normally, these registers store the recently issued CPU addresses during pipeline mode. To detect errors in these registers, the primary and duplicate address tag registers are continuously compared to each other if the buffer is valid. If they are different, then an address tag register error event is generated.

These registers are memory-mapped. All primary address tag registers are memory-mapped to one address and, likewise, all duplicate tag registers are mapped to another single address. During diagnostic mode, each individual set can be selected by the DIAG_BUF_SEL (Diagnostic Buffer Select) bit in the FDIAGCTRL register. User-supplied values can be written into the selected set during a diagnostic mode. If different values are written into the primary and the duplicate address tag registers, then the ADD_TAG_ERR (Address Tag Error) flag in the FEDACSTATUS register will be set. This diagnostic mode uses the FRAW_DATAL register to supply the alternate address when DIAG_TRIG is set. The FUNC_ERR_ADD register will not contain useful information during Diag mode 5. It will also trigger the normal uncorrectable register freeze.

All address tags and buffer valid bits will be cleared to 0 when leaving Diag mode 5. Going to mode 5 and back out clears the pipeline buffers and is useful for other test modes also. No functional checking is done by the address tag logic while the Diag mode is 5.

NOTE: The user should pre-load the registers with the test values with DIAG_TRIG = 0. After all test values are written, the DIAG_TRIG should then be set high to validate the diagnostic result.

4.6.2.6 ECC Data Correction Diagnostic Test Mode: DIAG_MODE = 7

Testing the error correction and ECC logic in the CPU involves corrupting the ECC value returned to the CPU. By inverting one or more bits of the ECC, the CPU will detect errors in a selected data or ECC bit, or in any possible value returned by the ECC.

To set an error for a particular bit, use the syndrome (see Section 4.3.3). For example, if you want to corrupt data bit 62 then put the value 5Bh into the test register.

The method uses the DATA_INV_PAR value in the FPAR_OVR register to alter the ECC during a slave access cycle. The value in the DATA_INV_PAR register is XORed with the current ECC to give a bad ECC value to the CPU. This only occurs when DIAG_MODE is 7, PAR_OVR_KEY in the FPAR_OVR register is 5h, DIAG_EN_KEY in the FDIAGCTRL register is 5h, and the access is a slave cycle.

This mode can set the FEDACSTATUS register status error bits B1_UNC_ERR or ERR_ZERO_FLG, but it will not set the D_UNC_ERR nor D_COR_ERR bits.

The sequence to do this test is:

1. Make sure the true DMA module is off.
2. Put 5h in BUS_PAR_DIS and 5h in PAR_OVR_KEY fields (00005Axxh) of the FPAR_OVR register.
3. Put the desired value in DAT_INV_PAR field of the FPAR_OVR register.
4. Put 7h in DIAG_MODE and 5h in DIAG_EN_KEY fields of the FDIAGCTRL register.
5. Read desired address from the mirrored Flash location. Mirrored Flash starts at address 0x20000000.
6. Put 0 in DIAG_MODE or Ah in one of the key fields to turn off this test.
7. Check error registers (FCOR_ERR_ADD, FEDACSTATUS, and FUNC_ERR_ADD) for ECC errors.
8. Repeat as necessary to test out the ECC.
9. Put 0 in DIAG_MODE field of the FDIAGCTRL register and Ah in both of the key fields to completely disable this test at the end of the test.
10. Put 2h in PAR_OVR_KEY field (00005400h) of the FPAR_OVR register to clear DAT_INV_PAR field.

4.6.3 Diagnostic Mode Summary

The following tables give a summary of the input registers needed for each mode, the possible registers that can change and the possible error bits in FEDACSTATUS that may set.

Table 4-9. Bus1 Diagnostic Mode Summary

DIAG MODE	Name	Inputs	Possible Outputs	Possible Error Bits Set	Notes
1	ECC Data Correction test mode				Not Applicable
2	ECC Syndrome Reporting test mode				Not Applicable
3	ECC Malfunction test mode 1				Not Applicable
4	ECC Malfunction test mode 2				Not Applicable
5	Address Tag Register test mode	FPRIM_ADD_TAG FDUP_ADD_TAG FRAW_DATAL	FUNC_ERR_ADD (1)	ADD_TAG_ERR	
6	Reserved				
7	ECC Data Correction Diagnostic test mode	DAT_INV_PAR	FUNC_ERR_ADD FCOR_ERR_ADD	B1_UNC_ERR ERR_ZERO_FLG	Slave access only

(1) Register output value changes, but does not contain useful information.

Table 4-10. Bus 2 and ECC Diagnostic Mode Summary

DIAG MODE	Name	Inputs	Possible Outputs	Possible Error Bits Set	Notes
1	ECC Data Correction test mode	FEUM_DMSW FEMU_DLSW FEMU_ECC FEMU_ADDR	FEMU_ECC FUNC_ERR_ADD FCOR_ERR_ADD FCOR_ERR_POS	D_UNC_ERR D_COR_ERR ERR_ONE_FLG ERR_ZERO_FLG ERR_PRF_FLG	
			FEMU_ECC EE_UNC_ERR_ADD EE_COR_ERR_ADD EE_COR_ERR_POS	EE_D_UNC_ERR EE_D_COR_ERR EE_ERR_ONE_FLG EE_ERR_ZERO_FLG EE_ERR_PRF_FLG	
2	ECC Syndrome Reporting test mode	FEMU_DMSW FEMU_DLSW FEMU_ECC FEMU_ADDR	FEMU_ECC	NA	
3	ECC Malfunction test mode 1	FRAW_DATAH FRAW_DATAL FRAW_ECC	FRAW_DATAH (1) FRAW_DATAL (1) FRAW_ECC (1) FUNC_ERR_ADD (1)	ECC_B2_MAL_ERR D_UNC_ERR	
			FRAW_DATAH (1) FRAW_DATAL (1) FRAW_ECC (1) EE_UNC_ERR_ADD (1)	EE_CME EE_D_UNC_ERR	
4	ECC Malfunction test mode 2	FRAW_DATAH FRAW_DATAL FRAW_ECC	FRAW_DATAH (2) FRAW_DATAL (2) FRAW_ECC (2) FUNC_ERR_ADD (2)	COMB2_MAL_G ECC_B2_MAL_ERR D_UNC_ERR	
			FRAW_DATAH (2) FRAW_DATAL (2) FRAW_ECC (2) EE_UNC_ERR_ADD (2)	EE_CMG EE_CME EE_D_UNC_ERR	
5	Address Tag Register test mode				Not applicable
6	Reserved				Not applicable
7	ECC Data Correction Diagnostic test mode				Not applicable

(1) Register output value will change, but will not contain useful information.
(2) Register output value will change, but will not contain useful information.

Table 4-11. Port Signals Diagnostic Mode Summary

DIAG MODE	Name	Error In	Uncorrectable Error ESM Group 3 Channel 7	Correctable Error ESM Group 1 Channel 6	Address Bus Parity Error ESM Group 2 Channel 4	FEE Uncorrectable Error ESM Group 1 Channel 36	FEE Correctable Error ESM Group 1 Channel 35
1	ECC Data Correction test mode	Bus 2	Yes	Yes	No	No	No
		EEPROM	No	No	No	Yes	Yes
2	ECC Syndrome Reporting test mode	Bus 2	No	No	No	No	No
		EEPROM	No	No	No	No	No
3	ECC Malfunction test mode 1	Bus 2	Yes	No	No	No	No
		EEPROM	No	No	No	Yes	No
4	ECC Malfunction test mode 2	Bus 2	Yes	No	No	No	No
		EEPROM	No	No	No	Yes	No
5	Address Tag Register test mode	Bus 1	Yes	No	No	No	No
6	Reserved	—	—	—	—	—	—
7	ECC Data Correction Diagnostic test mode	Bus 1	Yes	Yes	No	No	No

4.6.4 Read Margin

When the bits are programmed or erased, they are checked against a program_verify or erase_verify reference level that is far away from the normal read reference point. Over time, bit levels may drift toward the normal read point and if it is too much then a bit will read the wrong value. To counteract this, the bits can be read using different read_margin reference points to give an early detection of the problem. The bits can then be either re-programmed (most common) or the sector can be erased and reprogrammed.

4.7 Control Registers

This section details the Flash module registers, summarized in Table 4-12. A detailed description of each register and its bits is also provided.

The Flash module control registers can only be read and/or written by the CPU while in privileged mode. Each register begins on a word boundary. All registers are 32-bit, 16-bit and 8-bit accessible. The start address of the Flash module is FFF8 7000h.

Table 4-12. Flash Control Registers

Offset	Acronym	Register Description	Section
00h	FRDCNTL	Flash Option Control Register	Section 4.7.1
08h	FEDACTRL1	Flash Error Detection and Correction Control Register 1	Section 4.7.2
0Ch	FEDACTRL2	Flash Error Detection and Correction Control Register 2	Section 4.7.3
10h	FCOR_ERR_CNT	Flash Correctable Error Count Register	Section 4.7.4
14h	FCOR_ERR_ADD	Flash Correctable Error Address Register	Section 4.7.5
18h	FCOR_ERR_POS	Flash Correctable Error Position Register	Section 4.7.6
1Ch	FEDACSTATUS	Flash Error Detection and Correction Status Register	Section 4.7.7
20h	FUNC_ERR_ADD	Flash Un-Correctable Error Address Register	Section 4.7.8
24h	FEDACSDIS	Flash Error Detection and Correction Sector Disable Register	Section 4.7.9
28h	FPRIM_ADD_TAG	Flash Primary Address Tag Register	Section 4.7.10
2Ch	FDUP_ADD_TAG	Flash Duplicate Address Tag Register	Section 4.7.11
30h	FBPROT	Flash Bank Protection Register	Section 4.7.12
34h	FBSE	Flash Bank Sector Enable Register	Section 4.7.13
38h	FBBUSY	Flash Bank Busy Register	Section 4.7.14
3Ch	FBAC	Flash Bank Access Control Register	Section 4.7.15
40h	FBFALLBACK	Flash Bank Fallback Power Register	Section 4.7.16
44h	FBPRDY	Flash Bank/Pump Ready Register	Section 4.7.17
48h	FPAC1	Flash Pump Access Control Register 1	Section 4.7.18
4Ch	FPAC2	Flash Pump Access Control Register 2	Section 4.7.19
50h	FMAC	Flash Module Access Control Register	Section 4.7.20
54h	FMSTAT	Flash Module Status Register	Section 4.7.21
58h	FEMU_DMSW	EEPROM Emulation Data MSW Register	Section 4.7.22
5Ch	FEMU_DLSW	EEPROM Emulation Data LSW Register	Section 4.7.23
60h	FEMU_ECC	EEPROM Emulation ECC Register	Section 4.7.24
68h	FEMU_ADDR	EEPROM Emulation Address Register	Section 4.7.25
6Ch	FDIAGCTRL	Diagnostic Control Register	Section 4.7.26
70h	FRAW_DATAH	Uncorrected Raw Data High Register	Section 4.7.27
74h	FRAW_DATAL	Uncorrected Raw Data Low Register	Section 4.7.28
78h	FRAW_ECC	Uncorrected Raw ECC Register	Section 4.7.29
7Ch	FPAR_OVR	Parity Override Register	Section 4.7.30
C0h	FEDACSDIS2	Flash Error Detection and Correction Sector Disable Register 2	Section 4.7.31
288h	FSM_WR_ENA	FSM Register Write Enable	Section 4.7.32
2A4h	FSM_SECTOR	FSM Sector Register	Section 4.7.33
2B8h	EEPROM_CONFIG	EEPROM Emulation Configuration Register	Section 4.7.34
308h	EE_CTRL1	EEPROM Emulation Error Detection and Correction Control Register 1	Section 4.7.35
30Ch	EE_CTRL2	EEPROM Emulation Error Detection and Correction Control Register 2	Section 4.7.36
310h	EE_COR_ERR_CNT	EEPROM Emulation Correctable Error Count Register	Section 4.7.37
314h	EE_COR_ERR_ADD	EEPROM Emulation Correctable Error Address Register	Section 4.7.38
318h	EE_COR_ERR_POS	EEPROM Emulation Correctable Error Bit Position Register	Section 4.7.39
31Ch	EE_STATUS	EEPROM Emulation Error Status Register	Section 4.7.40
320h	EE_UNC_ERR_ADD	EEPROM Emulation Un-Correctable Error Address Register	Section 4.7.41
400h	FCFG_BANK	Flash Bank Configuration Register	Section 4.7.42

4.7.1 Flash Option Control Register (FRDCNTL)

FRDCNTL supports pipeline mode. This register controls Flash timings for the main Flash banks. For the equivalent register that controls Flash timings for the EEPROM Emulation Flash bank (bank 7), see Section 4.7.34.

Figure 4-8. Flash Option Control Register (FRDCNTL) [offset = 00h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved				RWAIT			
R-0				R/WP-1			

7	6	5	4	3	2	1	0
Reserved			ASWSTEN	Reserved			ENPIPE
R-0			R/WP-0	R-0			R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; *-n* = value after reset

Table 4-13. Flash Option Control Register (FRDCNTL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	RWAIT	0-Fh	Random/data Read Wait State The random read wait state bits indicate how many wait states are added to a Flash read access. In Pipeline mode there is always one wait state even when RWAIT is set to 0. **Note:** The required wait states for each HCLK frequency can be found in the device-specific data sheet.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	ASWSTEN		Address Setup Wait State Enable
		0	Address Setup Wait State is disabled.
		1	Address Setup Wait State is enabled. Address is latched one cycle before decoding to determine pipeline hit or miss. Address Setup Wait State is only available in pipeline mode. **Note:** The required address wait state for each HCLK frequency can be found in the device-specific data sheet.
3-1	Reserved	0	Reads return 0. Writes have no effect.
0	ENPIPE		Enable Pipeline Mode
		0	Pipeline mode is disabled.
		1	Pipeline mode is enabled.

4.7.2 Flash Error Detection and Correction Control Register 1 (FEDACCTRL1)

This register controls ECC event detection for the main Flash banks. For the equivalent register that controls ECC event detection for the EEPROM Emulation Flash bank (bank 7), see Section 4.7.35.

Figure 4-9. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) [offset = 08h]

31	25	24
Reserved		SUSP_IGNR
R-0		R/WP-0

23	20	19	16
Reserved		EDACMODE	
R-0		R/WP-Ah	

15	11	10	9	8
Reserved		EOFEN	EZFEN	EPEN
R-0		R/WP-0	R/WP-0	R/WP-0

7	4	3	0
Reserved		EDACEN	
R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-14. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	SUSP_IGNR		Suspend Ignore. In emulation mode, for example, viewing memory in the debugger's window, the CPU suspend signal is set. This bit determines whether the CPU suspend signal is ignored by the Flash module.
		0	CPU suspend signal blocks error bits setting and un-freezing. The Flash module blocks all errors from setting the error bits in emulation mode and blocks the un-freezing of the bits and registers by reading the FUNC_ERR_ADD register.
		1	CPU suspend has no effect on error bit setting and un-freezing. The Flash module ignores the CPU suspend signal and allows the error bits to set even in emulation mode. It also allows the Flash module to un-freeze the error bits and other registers by reading the FUNC_ERR_ADD register even in emulation mode.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	EDACMODE		Error Correction Mode for the main Flash banks. For EEPROM Emulation Flash bank (bank 7), see Section 4.7.35.
		5h	Single-bit errors during reads from OTP, ECC and the mirrored space (starting at 0x20000000) of banks 0 through 6, will be treated as uncorrectable errors by the Flash wrapper. The wrapper will assert an ESM group 3 error on channel 7 and the $\overline{\text{ERROR}}$ pin will be activated. No abort will be taken by the CPU.
		All other values	Single-bit errors during reads from OTP, ECC and the mirrored space (starting at 0x20000000) of banks 0 through 6, will be treated as correctable errors by the Flash wrapper. The wrapper will assert an ESM group 1 error on channel 6. The single-bit error will be corrected. **Note:** This mode does not affect reads from the main program Flash starting at address 0. **Note:** Reading ECC bits will generate an ECC error based on the contents of the 8 ECC bits and the 64 data bits they protect
15-11	Reserved	0	Reads return 0. Writes have no effect.

Table 4-14. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions (continued)

Bit	Field	Value	Description
10	EOFEN		Event on Ones Fail Enable
		0	No ESM error event is generated on a single-bit error where a 1 reads as a 0 when reading from the OTP or ECC memory locations.
		1	An ESM error event is generated on a single-bit error where a 1 reads as a 0 when reading from the OTP or ECC memory locations.
			Note: When either the EOFEN or the EZFEN bit is set, an error event will be generated on ESM group 1 channel 6 when any correctable error is generated by reading the main memory
9	EZFEN		Event on Zeros Fail Enable
		0	No ESM error event is generated on a single-bit error where a 0 reads as a 1 when reading from the OTP or ECC memory locations.
		1	An ESM error event is generated on a single-bit error where a 0 reads as a 1 when reading from the OTP or ECC memory locations.
			Note: When either the EOFEN or the EZFEN bit is set, an error event will be generated on ESM group 1 channel 6 when any correctable error is generated by reading the main memory
8	EPEN		Error Profiling Enable.
		0	Error profiling is disabled.
		1	Error profiling is enabled.
			The correctable error event is generated (ESM group 1 channel 6) when the number of CPU accesses of correctable bit errors detected and corrected has reached the threshold value defined in the FEDACCTRL2 register.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EDACEN		Error Detection and Correction Enable
		5h	CPU single and double error event signals are blocked.
			Note: It is recommended to enable ECC in the Flash wrapper by writing Ah to these bits before enabling ECC in the CPU. If ECC is enabled in the CPU, but not in the wrapper, the CPU will still check and correct single-bit ECC errors, and generate aborts on uncorrectable errors for the main Flash. However, the generation of ESM events, the capture of failing addresses and the detections and correction of errors in the OTP will be prevented.
		All other values	Error Detection and Correction events are captured and sent to the ESM

4.7.3 Flash Error Correction and Correction Control Register 2 (FEDACCTRL2)

This register applies to ECC event detection for the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 4.7.36.

Figure 4-10. Flash Error Correction and Correction Control Register 2 (FEDACCTRL2) [offset = 0Ch]

31	16
Reserved	
R-0	

15	0
SEC_THRESHOLD	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-15. Flash Error Correction Control and Correction Register 2 (FEDACCTRL2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	SEC_THRESHOLD	0-FFFFh	Single Error Correction Threshold When error profiling is enabled, this register contains the threshold value for the SEC (single error correction) occurrences before a correctable error event is generated (ESM group 1, channel 6). A threshold of 0 disables the threshold so that it does not generate an event.

4.7.4 Flash Correctable Error Count Register (FCOR_ERR_CNT)

This register applies to the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 4.7.37.

Figure 4-11. Flash Corectable Error Count Register (FCOR_ERR_CNT) [offset = 10h]

31	16
Reserved	
R-0	

15	0
FERRCNT	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-16. Flash Correctable Error Count Register (FCOR_ERR_CNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FERRCNT	0-FFFFh	Single Error Correction Count This register contains the number of SEC (single error correction) occurrences. Writing any value to this register resets the count value to 0. The counter resets to 0 when it increments to be equal to the single error correction threshold. This register only increments when profiling mode is enabled. This register is not affected by the EOFEN or EZEFEN error control bits in the FEDACCTRL1 register.

4.7.5 Flash Correctable Error Address Register (FCOR_ERR_ADD)

This register applies to the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 4.7.38.

The error address is captured during errors when either EOFEN or EZFEN enable bit is set. During error profiling mode when only EPEN is set, the error address is not captured if a correctable error is detected. This register is frozen while either the ERR_ZERO_FLG or the ERR_ONE_FLG bit is set in the FEDACSTATUS register.

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit to 1 in the FEDACCTRL1 register, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 4-12. Flash Correctable Error Address Register (FCOR_ERR_ADD) [offset = 14h]

31	16
COR_ERR_ADD	
R-u	

15	3	2	0
COR_ERR_ADD		B_OFF	
R-u		R-u	

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 4-17. Flash Correctable Error Address Register (FCOR_ERR_ADD) Field Descriptions

Bit	Field	Description
31-3	COR_ERR_ADD	Correctable Error Address COR_ERR_ADD records the CPU logical address of which a correctable error is detected by the ECC logic. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read.
2-0	B_OFF	Byte Offset Since ECC is checked on 64 bit data, when checking main memory or OTP, the address captured is aligned to a 64-bit boundary with address bits[2:0] equal to 0.When reading from the ECC bytes, these bits will indicate the failing address of the ECC location associated with the failure. When reading an ECC byte, the ECC is checked against the 64 data bits they protect.

4.7.6 Flash Correctable Error Position Register (FCOR_ERR_POS)

This register applies to the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 4.7.39.

Note: The bit error position is only detected during reads of the OTP, the mirrored Flash image or the ECC bytes. Single-bit errors corrected during reads of the main memory will only capture the failing address, but not the bit position. The bit position is captured during errors when either EOFEN or EZFEN enable bit is set. During error profiling mode when only EPEN is set, the bit position is not captured if a correctable error is detected. This register is frozen while either the ERR_ZERO_FLG or the ERR_ONE_FLG bit is set in the FEDACSTATUS register.

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit to 1 in the FEDACCTRL1 register, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 4-13. Flash Correctable Error Position Register (FCOR_ERR_POS) [offset = 18h]

31	16
Reserved	
R-0	

15	10	9	8	7	0
Reserved		BUS2	TYPE	ERR_POS	
R-0		R-u	R-u	R-u	

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 4-18. Flash Correctable Error Position Register (FCOR_ERR_POS) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	BUS2		Bus 2 Error
		0	The error was in the main Flash
		1	The error was from an OTP read
8	TYPE		ErrorType
		0	The error was one of the 64 data bits
		1	The error was one of the 8 check bits
7-0	ERR_POS	0-FFh	The bit address of the single-bit error.

4.7.7 Flash Error Detection and Correction Status Register (FEDACSTATUS)

This register applies to the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 4.7.40.

All these error status bits can be cleared by writing a 1 to the bit; writing a 0 has no effect.

The correctable errors in bits 2:0, and FSM_DONE bit 24, must be cleared before the end of their error event service routine or else the error event will re-issue.

Figure 4-14. Flash Error Detection and Correction Status Register (FEDACSTATUS) [offset = 1Ch]

31	25	24
Reserved		FSM_DONE
R-0		RCP-u

23	20	19	18	17	16
Reserved		COMB2_MAL_G	ECC_B2_MAL_ERR	B2_UNC_ERR	B2_COR_ERR
R-0		RCP-u	RCP-u	RCP-u	RCP-u

15	13	12	11	10	9	8
Reserved		D_UNC_ERR	ADD_TAG_ERR	ADD_PAR_ERR	Reserved	B1_UNC_ERR
R-0		RCP-u	RCP-u	RCP-u	R-0	RCP-u

7	4	3	2	1	0
Reserved		D_COR_ERR	ERR_ONE_FLG	ERR_ZERO_FLG	ERR_PRF_FLG
R-0		RCP-u	RCP-u	RCP-u	RCP-u

LEGEND: R = Read only; RCP = Read and Clear in Privilege Mode; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 4-19. Flash Error Detection and Correction Status Register (FEDACSTATUS) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	FSM_DONE		Flash State Machine Done This bit is set to 1 when the Flash state machine completes a program or erase operation. This bit will generate an interrupt on VIM channel 61 if the FSM_EVT_EN bit of the FSM_ST_MACHINE register is set. This bit must be cleared by writing a 1 to it in the interrupt routine to clear the interrupt request.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19	COMB2_MAL_G		Bus 2 Compare Malfunction Flag. 0 Compare Malfunction detected on the Bus2 SECDED in diagnostic mode 4 or not in diagnostic mode 1 Compare Malfunction not detected on the Bus2 SECDED or entered diagnostic mode 4 This bit becomes 1 when entering diagnostic mode 4, with DIAG_ECC_SEL field set to 0 or 1, and will be cleared if diagnostic mode 4 triggers an error. This bit will reset to 0 and will be 0 outside of diagnostic mode 4. Writing a 1 will set this bit to 1 only in diagnostic mode 4 otherwise writes have no effect.
18	ECC_B2_MAL_ ERR		Bus 2 ECC Malfunction Error Flag 0 SECDED malfunction not detected on Bus2 1 SECDED malfunction detected on Bus2
17	B2_UNC_ERR		Bus 2 uncorrectable error 0 No bus 2 uncorrectable errors were detected 1 A bus 2 uncorrectable error was detected Two or more bits in the data, or ECC field; or a single-bit error in the address field have been found in error. Address-bit errors are considered an uncorrectable error. The FUNC_ERR_ADD register should contain the Bus2 error location. This error will generate an ESM group 3 channel 7 event.

Table 4-19. Flash Error Detection and Correction Status Register (FEDACSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
16	B2_COR_ERR		Bus 2 Correctable Error
		0	No bus2 correctable error was detected
		1	A bus 2 correctable error was detected
			One bit in the data, or ECC field has been found in error. Either the ERR_ONE_FLAG or ERR_ZERO_FLAG should be set in this register along with this bit. The FCOR_ERR_ADD register should contain the error address, and the FCOR_ERR_POS register should contain the failing bit position. This error will generate an ESM group 1 channel 6 event.
15-13	Reserved	0	Reads return 0. Writes have no effect.
12	D_UNC_ERR		Diagnostic Uncorrectable Error
			This bit sets when diagnostic mode 1 discovers a multi-bit error using he ECC. This means two or more bits in the data, address or ECC field have been found in error. The ECC is capable of correcting a single-bit error and this would show up in bit 3 D_COR_ERR. The ECC can always detect two bit errors. Three or more bit errors may escape detection with the ECC. This bit also may set during other uncorrectable errors and during the diagnostic mode like address tag errors and ECC malfunctions.
11	ADD_TAG_ERR		Address Tag Register Error Flag
		0	Address Tag Register Error not detected
		1	Address Tag Register Error detected.
			This bit is set if the primary address tag has a hit but the duplicate address tag does not match the primary address tag. This bit is functional only when pipeline mode is enabled. This error will create an ESM group 3 channel 7 event.
10	ADD_PAR_ERR		Address Parity Error Flag
		0	No address parity error was detected.
		1	A parity error was detected on the incoming address bus.
			The full 32 bit address will be stored in FUNC_ERR_ADD register. This error will create an ESM group 2 channel 4 event.
9	Reserved	0	Reads return 0. Writes have no effect.
8	B1_UNC_ERR		Bus 1 Uncorrectable Error Flag
		0	No Bus 1 uncorrectable errors were detected
		1	A bus 1 uncorrectable error was detected
			Two or more bits in the data, or ECC field; or a single-bit error in the address field have been found in error. Address-bit errors are considered an uncorrectable error. The FUNC_ERR_ADD register will contain the Bus1 error location. This error will generate an ESM group 3 channel 7 event.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3	D_COR_ERR		Diagnostic Correctable Error Status Flag
			This bit sets when diagnostic mode 1 discovers a single-bit correctable error using the ECC. Multi-bit errors are flagged using the D_UNC_ERR bit. The uncorrectable error address must be unfrozen in order to set this bit.
2	ERR_ONE_FLG		Error on One Fail Status Flag
		0	No correctable error where a 1 read as a 0 on bus 2
		1	A correctable error occurred on bus 2 where a 1 read as a 0
			This bit is set if the EOFEN (Error on One Fail Enable) bit is set then, and one bit in the data, or ECC field that should have been read as a 1, read as a 0. During the read the bit is corrected to a 1. The FCOR_ERR_ADD register will contain the bus 2 error address, and the FCOR_ERR_POS register will contain the failing bit position. This error will generate an ESM group 1 channel 6 event. When this bit is set, the B2_CORR_ERR bit will also be set. This error will generate an ESM group 1 channel 6 event.

Table 4-19. Flash Error Detection and Correction Status Register (FEDACSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
1	ERR_ZERO_ FLG		Error on Zero Fail Status Flag
		0	No correctable errors on bus1 nor any correctable errors on bus 2 where a 0 was read as a 1
		1	A correctable error occurred on bus 1, or a correctable error occurred on bus 2 where a 0 was read as a 1
			This bit is set if the EZFEN (Error on Zero Fail Enable) bit is set and a correctable error is detected on bus 2 where a 0 is read as a 1 and corrected to be a 0, or if either the EZFEN or the EOFEN bits are set and any single-bit error is detected and corrected on bus 1. The FCOR_ERR_ADD register will contain the error address. If the error was on bus 2, then the B2_COR_ERR bit will also be set and the FCOR_ERR_POS register will contain the failing bit position. The FCOR_ERR_POS register will not indicate the failing bit position for a bus 1 error. This error will generate an ESM group 1 channel 6 event.
0	ERR_PRF_FLG		Error Profiling Status Flag
		0	Error profiling is not enabled, or the number of correctable errors has not reached the threshold programmed into the SEC_THRESHOLD register
		1	Error profiling is enabled and the number of correctable errors has reached the threshold programmed into the SEC_THRESHOLD register

4.7.8 Flash Uncorrectable Error Address Register (FUNC_ERR_ADD)

This register applies to ECC event detection for the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 4.7.41.

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit to 1 in the FEDACCTRL1 register, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 4-15. Flash Uncorrectable Error Address Register (FUNC_ERR_ADD) [offset = 20h]

31	16
UNC_ERR_ADD	
R-u	

15	3	2	0
UNC_ERR_ADD		B_OFF	
R-u		R-u	

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 4-20. Flash Uncorrectable Error Address Register (FUNC_ERR_ADD) Field Descriptions

Bit	Field	Description
31-3	UNC_ERR_ADD	Uncorrectable Error Address UNC_ERR_ADD records the CPU logical address of which an uncorrectable error is detected by the ECC logic in the CPU. The UNC_ERR_ADD also captures the error address when a address bus parity mismatch is detected. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read. This register captures the full 32 bit incoming address when there is a bus parity error. It only captures address of 22:3 for multiple bit ECC errors. Address parity errors take priority over other errors that happen in the same cycle.
2-0	B_OFF	Byte offset Since ECC is checked on 64 bit data, when checking main memory or OTP, the address captured is aligned to a 64-bit boundary with address bits[2:0] equal to 0.When reading from the ECC bytes, these bits will indicate the failing address of the ECC location associated with the failure. When reading an ECC byte, the ECC is checked against the 64 data bits they protect.

4.7.9 Flash Error Detection and Correction Sector Disable Register (FEDACSDIS)

This register is used to disable the SECDED function for one or two sectors from the EEPROM Emulation Flash (bank 7). An additional two sectors can have SECDED disabled by the use of the FEDACSDIS2 register (see Section 4.7.31).

Figure 4-16. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS) [offset = 24h]

31 29	28	27 24	23 21	20	19 16
BankID1_Inverse	Rsvd	SectorID1_inverse	BankID1	Rsvd	SectorID1
RWP-0	R-0	RWP-0	RWP-0	R-0	RWP-0

15 13	12	11 8	7 5	4	3 0
BankID0_Inverse	Rsvd	SectorID0_inverse	BankID0	Rsvd	SectorID0
RWP-0	R-0	RWP-0	RWP-0	R-0	RWP-0

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-21. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS) Field Descriptions

Bit	Field	Value	Description
31-29	BankID1_Inverse		The bank ID inverse bits are used with the bank ID bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		0	If BankID1 = 7h and BankID1_inverse = 0, then if a valid sector is selected by SectorID1 and SectorID1_inverse that sector will have ECC checking disabled.
		All other values	No sector is disabled by disable ID 1.
28	Reserved	0	Read returns 0. Writes have no effect.
27-24	SectorID1_inverse	0-Fh	The sector ID inverse bits are used with the sector ID bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 1.
23-21	BankID1		The bank ID bits are used with the bank ID inverse bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		7h	If BankID1 = 7h and BankID1_inverse = 0, then if a valid sector is selected by SectorID1 and SectorID1_inverse that sector will have ECC checking disabled.
		All other values	No sector is disabled by disable ID 1.
20	Reserved	0	Read returns 0. Writes have no effect.
19-16	SectorID1	0-Fh	The sector ID bits are used with the sector ID inverse bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 1.
15-13	BankID0_Inverse		The bank ID inverse bits are used with the bank ID bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		0	If BankID0 = 7h and BankID0_inverse = 0, then if a valid sector is selected by SectorID0 and SectorID0_inverse that sector will have ECC checking disabled.
		All other values	No sector is disabled by disable ID 0.
12	Reserved	0	Read returns 0. Writes have no effect.
11-8	SectorID0_inverse	0-Fh	The sector ID inverse bits are used with the sector ID bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 0.
7-5	BankID0		The bank ID bits are used with the bank ID inverse bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		7h	If BankID0 = 7h and BankID0_inverse = 0, then if a valid sector is selected by SectorID0 and SectorID0_inverse that sector will have ECC checking disabled.
		All other values	No sector is disabled by disable ID 0.

Table 4-21. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS) Field Descriptions (continued)

Bit	Field	Value	Description
4	Reserved	0	Read returns 0. Writes have no effect.
3-0	SectorID0	0-Fh	The sector ID bits are used with the sector ID inverse bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 0.

4.7.10 Primary Address Tag Register (FPRIM_ADD_TAG)

This register is used to test the pipeline address tag registers (see Section 4.6.2.5).

Figure 4-17. Primary Address Tag Register (FPRIM_ADD_TAG) [offset = 28h]

31	16
PRIM_ADD_TAG[31:16]	
RWP-0	

15	4	3	0
PRIM_ADD_TAG[15:4]		0000	
RWP-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode;; -*n* = value after reset;

Table 4-22. Primary Address Tag Register (FPRIM_ADD)_TAG Field Descriptions

Bit	Field	Value	Description
31-4	PRIM_ADD_TAG	0-FFF FFFFh	Primary Address Tag Register The primary address tag register selected by the DIAG_BUF_SEL bits in the FDIAGCTRL register is memory-mapped here. This register can only be written in privileged mode when diagnostic mode is enabled with DIAG_EN_KEY = 5h and DIAG_MODE = 5h in the FDIAGCTRL register. This register is not updated with new Flash data if DIAG_EN_KEY is not equal to 5h or DIAG_MODE is 0 or 7h. Valid reads can occur in any mode. The register clears when an address tag error is found and when leaving DIAG_MODE 5.
3-0	Reserved	0	Reads return 0. Writes have no effect.

4.7.11 Duplicate Address Tag Register (FDUP_ADD_TAG)

This register is used to test the pipeline address tag registers (see Section 4.6.2.5).

Figure 4-18. Duplicate Address Tag Register (FDUP_ADD_TAG) [offset = 2Ch]

31	16
DUP_ADD_TAG[31:16]	
RWP-0	

15	4	3	0
DUP_ADD_TAG[15:4]		0000	
RWP-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode;; -*n* = value after reset;

Table 4-23. Duplicate Address Tag Register (FDUP_ADD)_TAG Field Descriptions

Bit	Field	Value	Description
31-4	DUP_ADD_TAG	0-FFF FFFFh	Primary Address Tag Register The duplicate address tag register selected by the DIAG_BUF_SEL bits in the FDIAGCTRL register is memory-mapped here. This register can only be written in privileged mode when diagnostic mode is enabled with DIAG_EN_KEY = 5h and DIAG_MODE = 5h in the FDIAGCTRL register. This register is not updated with new Flash data if DIAG_EN_KEY is not equal to 5h or DIAG_MODE is 0 or 7h. Valid reads can occur in any mode. The register clears when an address tag error is found and when leaving DIAG_MODE 5.
3-0	Reserved	0	Reads return 0. Writes have no effect.

4.7.12 Flash Bank Protection Register (FBPROT)

Figure 4-19. Flash Bank Protection Register (FBPROT) [offset = 30h]

31	16
Reserved	
R-0	

15	1	0
Reserved		PROTL1DIS
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-24. Flash Bank Protection Register (FBPROT) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	PROTL1DIS		Level 1 Protection Disable bit Setting this bit disables protection from writing to the OTPPROTDIS bits in the FBAC register as well as the BSE bits for all banks in the FBSE register. Clearing this bit enables protection and disables write access to the OTPPROTDIS bits and FBSE register.
		0	Level 1 protection is enabled.
		1	Level 1 protection is disabled.

4.7.13 Flash Bank Sector Enable Register (FBSE)

FBSE provides one enable bit per sector for up to 16 sectors per bank. Each bank in the Flash module has one FBSE register. The bank is selected via the BANK bits in the FMAC register. As only one bank at a time can be selected by FMAC, only the register for the bank selected appears at this address.

Figure 4-20. Flash Bank Sector Enable Register (FBSE) [offset = 34h]

31	16
Reserved	
R-0	

15	0
BSE[15:0]	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-25. Flash Bank Sector Enable Register (FBSE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BSE		Bank Sector Enable Each bit corresponds to a Flash sector in the bank specified by the FMAC register. Bit 0 corresponds to sector 0, bit 1 corresponds to sector 1, and so on. These bits can be set only when PROTL1DIS = 1 in the FBPROT register and in privilege mode.
		0	The corresponding numbered sector is disabled for program or erase access.
		1	The corresponding numbered sector is enabled for program or erase access.

4.7.14 Flash Bank Busy Register (FBBUSY)

Figure 4-21. Flash Bank Busy Register (FBBUSY) [offset = 38h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		BUSY[7:0]	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 4-26. Flash Bank Busy Register (FBBUSY) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	BUSY		Bank Busy. Each bit corresponds to a Flash bank.
		0	The corresponding bank is not busy
		1	The corresponding bank is busy with a state machine or bus 2 operation, or the bank is not implemented

4.7.15 Flash Bank Access Control Register (FBAC)

Figure 4-22. Flash Bank Access Control Register (FBAC) [offset = 3Ch]

31	24	23	16
Reserved		OTPPROTDIS[7:0]	
R-0		R/WP-0	

15	8	7	0
BAGP		VREADST	
R/WP-0		R/WP-Fh	

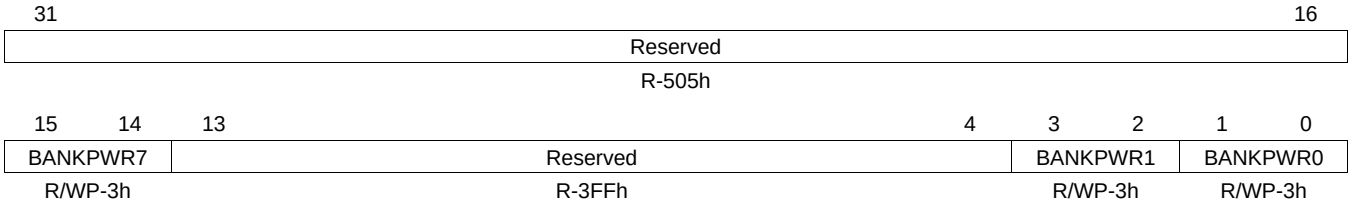
LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-27. Flash Bank Access Control Register (FBAC) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	OTPPROTDIS		OTP Sector Protection Disable. Each bit corresponds to a Flash bank. This bit can be set only when PROTL1DIS = 1 in the FBPROT register and in privilege mode.
		0	Programming of the OTP sector is disabled.
		1	Programming of the OTP sector is enabled.
15-8	BAGP	0-FFh	Bank Active Grace Period. These bits contain the starting count value for the BAGP down counter. Any access to a given bank causes its BAGP counter to reload the BAGP value for that bank. After the last access to this Flash bank, the down counter delays from 0 to 255 prescaled HCLK clock cycles before putting the bank into one of the fallback power modes as determined by the FBFALLBACK register. This value must be greater than 1 when the fallback mode is not ACTIVE. **Note:** The prescaled clock used for the BAGP down counter is a clock divided by 16 from HCLK.
7-0	VREADST	0-FFh	VREAD Setup. VREAD is generated by the Flash pump and used for Flash read operation. The bank power up sequencing starts VREADST HCLK cycles after VREAD power supply becomes stable. **Note**: There is not a programmable Bank Sleep counter and Standby counter register. The number of clock cycles to transition from sleep to standby and standby to active is hardcoded in the Flash wrapper design.

4.7.16 Flash Bank Fallback Power Register (FBFALLBACK)

Figure 4-23. Flash Bank Fallback Power Register (FBFALLBACK) [offset = 40h]



LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-28. Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	505h	Do not write to these register bits
15-14	BANKPWR7		Bank 7 Fallback Power Mode.
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode
13-4	Reserved	3FFh	Do not write to these register bits.
3-2	BANKPWR1		Bank 1 Fallback Power Mode.
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode
1-0	BANKPWR0		Bank 0 Fallback Power Mode.
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode

4.7.17 Flash Bank/Pump Ready Register (FBPRDY)

Figure 4-24. Flash Bank/Pump Ready Register (FBPRDY) [offset = 44h]

31	24	23	16
Reserved		BANKBUSY[n]	
R-0		R-1 (Unimplemented banks) or R-0 (Implemented banks)	

15	14	8	7	0
PUMPRDY	Reserved		BANKRDY[n]	
R-1	R-0		R-1	

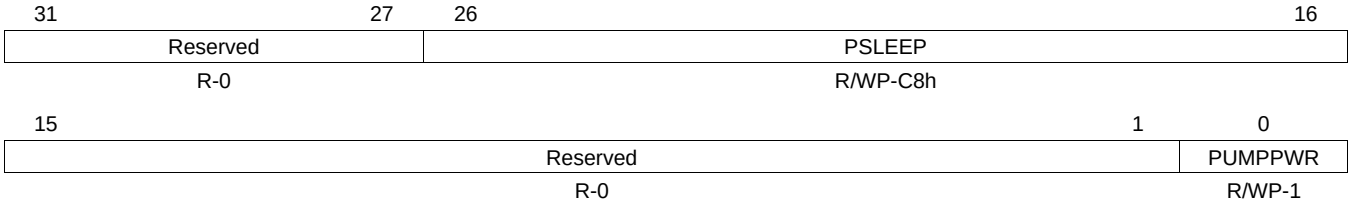
LEGEND: R = Read only; -*n* = value after reset

Table 4-29. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Read returns 0. Writes have no effect.
23-16	BANKBUSY[7:0]		Bank busy bits (one bit for each bank)
		0	The bank is not busy.
		1	The bank is busy, not ready or this bank is not implemented
			Note: A bank is considered busy if it is being accessed by the TCM, Bus2 or the Flash state machine.
15	PUMPRDY		Flash pump ready flag
		0	Pump is not ready (Code must be executing from somewhere other than internal Flash)
		1	Pump is ready for Flash accesses
14-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	BANKRDY[7:0]		Bank ready bits (one bit for each bank)
		0	Flash bank is in the sleep or standby state
		1	Flash bank is in the active state, or the bank is not implemented

4.7.18 Flash Pump Access Control Register 1 (FPAC1)

Figure 4-25. Flash Pump Access Control Register 1 (FPAC1) [offset = 48h]



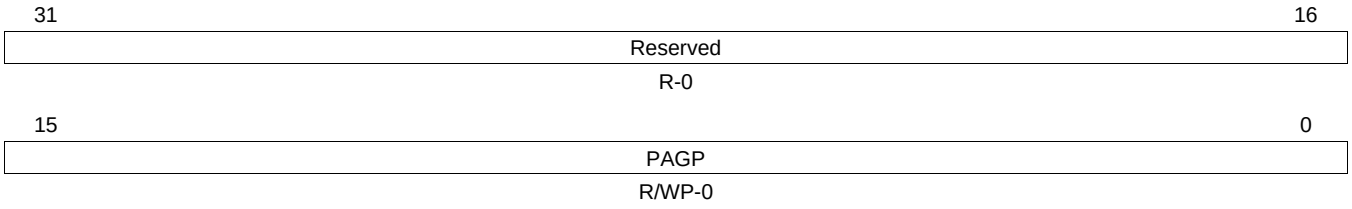
LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-30. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
26-16	PSLEEP	0-7FFh	Pump Sleep. These bits contain the starting count value for the charge pump sleep down counter. While the charge pump is in sleep mode, the power mode management logic holds the charge pump sleep counter at this value. When the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP pump sleep down clock cycles before putting the charge pump into active power mode. **Note:** Pump sleep down counter clock is a divide by 2 input of HCLK. That is, there are 2 × HCLK cycles for every PSLEEP counter cycle.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	PUMPPWR		Flash Charge Pump Fallback Power Mode
		0	Sleep (all pump circuits disabled)
		1	Active (all pump circuits active)

4.7.19 Flash Pump Access Control Register 2 (FPAC2)

Figure 4-26. Flash Pump Access Control Register 2 (FPAC2) [offset = 4Ch]



LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-31. Flash Pump Access Control Register 2 (FPAC2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	PAGP	0-FFFFh	Pump Active Grace Period This register contains the starting count value for the PAGP mode down counter. Any access to Flash memory causes the counter to reload with the PAGP value. After the last access to Flash memory, the down counter delays from 0 to 65535 prescaled HCLK clock cycles before entering one of the charge pump fallback power modes as determined by PUMPPWR in the FPAC1 register. **Note:** The PAGP down counter is clocked by the same prescaled clock as the BAGP down counter that is a divide by 16 of HCLK.

4.7.20 Flash Module Access Control Register (FMAC)

Figure 4-27. Flash Module Access Control Register (FMAC) [offset = 50h]

31	16
Reserved	
R-0	

15	3	2	0
Reserved		BANK	
R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-32. Flash Module Access Control Register (FMAC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
2-0	BANK	0-7h	Bank Enable. These bits select which bank is enabled for operations such as local register access, OTP sector access, and program/erase commands. These bits select only one bank at a time from up to eight banks depending on the specific device being used. For example, a 000 selects bank 0; 011 selects bank 3. **Note:** BANK can identify up to 8 Flash banks. If BANK is selected for an un-implemented bank, then the BANK will set itself to the number of an implemented bank. To determine if a bank is implemented, write the bank number to BANK and read back the value to see if what was written can be read back.

4.7.21 Flash Module Status Register (FMSTAT)

Figure 4-28. Flash Module Status Register (FMSTAT) [offset = 54h]

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	ILA	Reserved	PGV	Reserved	EV	Reserved	BUSY
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ERS	PGM	INV-DAT	CSTAT	VOLTSTAT	ESUSP	PSUSP	SLOCK
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 4-33. Flash Module Status Register (FMSTAT) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Read returns 0. Writes have no effect.
14	ILA		Illegal Address When set, indicates that an illegal address is detected. Three conditions can set illegal address flag. 1. Writing to a hole (un-implemented logical address space) within a Flash bank. 2. Writing to an address location to an un-implemented Flash space. 3. Input address for write is decoded to select a different bank from the bank ID register. 4. The address range does not match the type of FSM command. For example, the erase_sector command must match the address regions. 5. TI-OTP address selected but CMD_EN in FSM_ST_MACHINE is not set.
13	Reserved	0	Read returns 0. Writes have no effect.
12	PGV		Program Verify When set, indicates that a word is not successfully programmed after the maximum allowed number of program pulses are given for program operation.
11	Reserved	0	Read returns 0. Writes have no effect.
10	EV		Erase Verify When set, indicates that a sector is not successfully erased after the maximum allowed number of erase pulses are given for erase operation. During Erase verify command, this flag is set immediately if a bit is found to be 0.
9	Reserved	0	Read returns 0. Writes have no effect.
8	BUSY		When set, this bit indicates that a program, erase, or suspend operation is being processed.
7	ERS		Erase Active When set, this bit indicates that the Flash module is actively performing an erase operation. This bit is set when erasing starts and is cleared when erasing is complete. It is also cleared when the erase is suspended and set when the erase resumes.
6	PGM		Program Active When set, this bit indicates that the Flash module is currently performing a program operation. This bit is set when programming starts and is cleared when programming is complete. It is also cleared when programming is suspended and set when programming is resumes.
5	INVDAT		Invalid Data When set, this bit indicates that the user attempted to program a “1” where a “0” was already present. This bit is cleared by the Clear Status command.
4	CSTAT		Command Status Once the FSM starts any failure will set this bit. When set, this bit informs the host that the program, erase, or validate sector command failed and the command was stopped. This bit is cleared by the Clear Status command. For some errors, this will be the only indication of an FSM error because the cause does not fall within the other error bit types.

Table 4-33. Flash Module Status Register (FMSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
3	VOLTSTAT		Core Voltage Status When set, this bit indicates that the core voltage generator of the pump power supply dipped below the lower limit allowable during a program or erase operation. This bit is cleared by the Clear Status command.
2	ESUSP		Erase Suspended When set, this bit indicates that the Flash module has received and processed an erase suspend operation. This bit remains set until the erase resume command has been issued or until the Clear_More command is run.
1	PSUSP		Program Suspended When set, this bit indicates that the Flash module has received and processed a program suspend operation. This bit remains set until the program resume command has been issued or until the Clear_More command is run.
0	SLOCK		Sector Lock Status When set, this bit indicates that the operation was halted because the target sector was locked for erasing and programming either by the sector protect bit or by OTP write protection disable bits. (Bits BSE in FBSE register or OTPPROTDIS in register FBAC). This bit is cleared by the Clear Status command. No SLOCK FSM error will occur if all sectors in a bank erase operation are set to 1. All the sectors will be checked but no SLOCK will be set if no operation occurs due to the SECT_ERASED bits being set to all 1s. A SLOCK error will occur if attempting to do a sector erase with either BSE is cleared or SECT_ERASED is set.

4.7.22 EEPROM Emulation Data MSW Register (FEMU_DMSW)

The Flash module controller includes hardware support computing the check bits for ECC, based on the data and address being programmed into the EEPROM emulation array. To utilize this capability, the address to be programmed is written to the FEMU_ADDR register and the data to be programmed is written to the FEMU_DMSW and FEMU_DLSW registers. The write to FEMU_DLSW triggers an ECC calculation and the resulting check bits are available in the FEMU_ECC register. The value from FEMU_ECC can then be used to program the check bits into the EEPROM emulation array for the particular data word over which they were calculated.

Figure 4-29. EEPROM Emulation Data MSW Register (FEMU_DMSW) [offset = 58h]

31	0
EMU_DMSW[63:32]	
RWP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; -*n* = value after reset

Table 4-34. EEPROM Emulation Data MSW Register (FEMU_DMSW) Field Descriptions

Bit	Field	Description
31-0	EMU_DMSW	EEPROM Emulation Most-Significant Data Word. The most-significant data word of the 64-bits of data for which the ECC check bits are to be calculated should be programmed into this register. This register is also used in diagnostic modes 1 and 2 where it supplies the upper data for checking the SECDED hardware.

4.7.23 EEPROM Emulation Data LSW Register (FEMU_DLSW)

The Flash module controller includes hardware support computing the check bits for ECC, based on the data and address being programmed into the EEPROM emulation array. To utilize this capability, the address to be programmed is written to the FEMU_ADDR register and the data to be programmed is written to the FEMU_DMSW and FEMU_DLSW registers. The write to FEMU_DLSW triggers an ECC calculation and the resulting check bits are available in the FEMU_ECC register. The value from FEMU_ECC can then be used to program the check bits into the EEPROM emulation array for the particular data word over which they were calculated.

Figure 4-30. EEPROM Emulation Data LSW Register (FEMU_DLSW) [offset = 5Ch]

31	0
EMU_DLSW[31:0]	
RWP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; -*n* = value after reset

Table 4-35. EEPROM Emulation Data LSW Register (FEMU_DLSW) Field Descriptions

Bit	Field	Description
31-0	EMU_DLSW	EEPROM Emulation Least-Significant Data Word. The least-significant data word of the 64-bits of data for which the ECC check bits are to be calculated should be programmed into this register. This register is also used in diagnostic modes 1 and 2 where it supplies the lower data for checking the SECDED hardware.

4.7.24 EEPROM Emulation ECC Register (FEMU_ECC)

NOTE: This register is only available when the module is configured to use the ECC logic; otherwise, this register is reserved.

The Flash module controller includes hardware support computing the check bits for ECC, based on the data and address being programmed into the EEPROM emulation array. To utilize this capability, the address to be programmed is written to the FEMU_ADDR register and the data to be programmed is written to the FEMU_DMSW and FEMU_DLSW registers. The write to FEMU_DLSW triggers an ECC calculation and the resulting check bits are available in the FEMU_ECC register. The value from FEMU_ECC can then be used to program the check bits into the EEPROM emulation array for the particular data word over which they were calculated.

Figure 4-31. EEPROM Emulation ECC Register (FEMU_ECC) [offset = 60h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		EMU_ECC[7:0]	
R-0		R/WP-3h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; -*n* = value after reset

Table 4-36. EEPROM Emulation ECC Register (FEMU_ECC) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	EMU_ECC		EEPROM Emulation ECC Check Bit Value. This register contains the ECC check bits calculated by the FMC controller based on the address written to FEMU_ADDR and the 64-bits of data written to FEMU_DMSW and FEMU_DLSW. This register is also used in the diagnostic modes 1 and 2. In these modes, this register supplies the ECC data for checking the SECDED. In mode 1, this register is filled with the desired ECC and after the DIAG_TRIG is set this register is set to the ECC value returned from the SECDED selected by DIAG_ECC_SEL. DIAG_EN_KEY and DIAG_TRIG must be set to fill this register in the diagnostic modes. Writes to FEMU_DxSW will not affect this register in diagnostic modes 1 or 2. In mode 2, this register is filled with the desired ECC and after the DIAG_TRIG is set this register is set to the syndrome value returned from the SECDED selected by DIAG_ECC_SEL. This register is only available when the module is configured to use the ECC logic; otherwise, it is a reserved register.

4.7.25 EEPROM Emulation Address Register (FEMU_ADDR)

The Flash module controller includes hardware support computing the check bits for ECC, based on the data and address being programmed into the EEPROM emulation array. To utilize this capability, the address to be programmed is written to the FEMU_ADDR register and the data to be programmed is written to the FEMU_DMSW and FEMU_DLSW registers. The write to FEMU_DLSW triggers an ECC calculation and the resulting check bits are available in the FEMU_ECC register. The value from FEMU_ECC can then be used to program the check bits into the EEPROM emulation array for the particular data word over which they were calculated.

Figure 4-32. EEPROM Emulation Address Register (FEMU_ADDR) [offset = 68h]

31	22	21	16
Reserved		EMU_ADDR[21:16]	
R-0		RWP-0	

15	3	2	1	0
EMU_ADDR[15:3]		Reserved		
RWP-0		R-0		

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; -*n* = value after reset

Table 4-37. EEPROM Emulation Address Register (FEMU_ADDR) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Reserved. Writes have no effect. It is not necessary to mask these upper ten bits when writing the address of bank 7 locations (0xF002xxxx); however, these bits are not used in calculating the ECC value and will read back as 0.
21-3	EMU_ADDR	0-7 FFFFh	EEPROM Emulation Address. The address of the 64-bit data word over which ECC is to be calculated is written to this field. Note that only bits 21:3 are actually written and used for the calculation. The other bits (31:22 and 2:0) are ignored, but do not need to be masked off before being written to this register. This register is also used in diagnostic modes 1 and 2 where it supplies the address bits for checking the SECDED hardware.
2-0	Reserved	0	Reserved. Writes have no effect. The lower three bits of the CPU address are not used in the ECC calculation to align the data on a 64-bit boundary. These bits will read back as 0.

4.7.26 Diagnostic Control Register (FDIAGCTRL)

First set the DIAG_MODE and the DIAG_EN_KEY bits before setting up the other registers to block the other registers from causing a false error. The final write should set the DIAG_TRIG bit to activate the test. Running out of RAM will prevent problems with the diagnostic test corrupting the Flash access in some of the modes.

Figure 4-33. Diagnostic Control Register (FDIAGCTRL) [offset = 6Ch]

31	25	24	23	20	19	16
Reserved		DIAG_TRIG	Reserved		DIAG_EN_KEY	
R-0		R/WP-0	R-0		R/WP-Ah	

15	14	12	11	10	9	8	7	3	2	0
Rsvd	DIAG_ECC_SEL		Reserved		DIAG_BUF_ SEL		Reserved		DIAG_MODE	
R-0	R/WP-0		R-0		R/WP-0		R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; -*n* = value after reset

Table 4-38. Diagnostic Control Register (FDIAGCTRL) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24	DIAG_TRIG		Diagnostic Trigger Diagnostic trigger is the final qualifier for the diagnostic result. After setting all the other diagnostic register values, the DIAG_TRIG is set to 1. This activates the diagnostic logic for one access and then automatically clears the DIAG_TRIG value. The DIAG_EN_KEY and DIAG_MODE bits must be set at least one cycle before setting DIAG_TRIG. This bit always reads as 0.
23-20	Reserved	0	Reserved
19-16	DIAG_EN_KEY		Diagnostic Enable Key
		5h	Diagnostic mode is enabled.
		All other values	Diagnostic mode is disabled.
15	Reserved	0	Reserved
14-12	DIAG_ECC_SEL		Diagnostic SECDED Select
		0	Select SECDED0 for diagnostic testing
		1h	Select SECDED1 for diagnostic testing
		2h	Select SECDED2 for diagnostic testing (256 bit wide words only)
		3h	Select SECDED3 for diagnostic testing (256 bit wide words only
		4h	Select BUS2 SECDED for diagnostic testing
		5h	Select FEE SECDED for diagnostic testing (same ECC logic as BUS2, but sets the FEE registers)
		6h-7h	Reserved
11-10	Reserved	0	Reserved
9-8	DIAG_BUF_ SEL		Diagnostic Buffer Select
		0	The DIAG_BUF_SEL selects the Instruction or Data buffer to read or write when accessing the FPRIM_ADD_TAG and FDUP_ADD_TAG registers. The address tags consists of matching primary and duplicate address tag registers. All the primary address tag registers are memory mapped to a common address (see Section 4.7.10) and are selected by DIAG_BUF_SEL. The same occurs for the duplicate address (see Section 4.7.11). Bit 0 selects a data buffer if high and an instruction buffer if low. Bit 1 indicates the buffer number.
		0	Instruction Buffer 0
		1h	Data Buffer 0
		2h	Instruction Buffer 1
		3h	Data Buffer 1

Table 4-38. Diagnostic Control Register (FDIAGCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	Reserved	0	Reserved
2-0	DIAG_MODE		Diagnostic Mode
		0	Diagnostic mode is disabled. This is the same as DIAG_EN_KEY is not equal to 5h.
		1h	Diagnostic ECC Data Correction test mode (see Section 4.6.2.1)
		2h	Diagnostic ECC Syndrome Reporting test mode (see Section 4.6.2.2)
		3h	ECC Malfunction test mode 1 (same data) (see Section 4.6.2.3)
		4h	ECC Malfunction test mode 2 (inverted data) (see Section 4.6.2.4)
		5h	Address Tag Register test mode (see Section 4.6.2.5)
		6h	Reserved
		7h	ECC Data Correction Diagnostic test mode (see Section 4.6.2.6)

4.7.27 Uncorrected Raw Data High Register (FRAW_DATAH)

NOTE: Raw Data and Raw ECC registers can be loaded with diagnostic values only in diagnostic modes 1 through 6 with DIAG_EN_KEY = 5h in the Diagnostic Control Register (FDIAGCTRL). These modes must be set for at least one clock cycle before writing to any FRAW* register.

Figure 4-34. Uncorrected Raw Data High Register (FRAW_DATAH) [offset = 70h]

31	0
RAW_DATA[63:32]	
RWP-u	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; -*n* = value after reset; -u = Unchanged value on internal reset, cleared on power up

Table 4-39. Uncorrected Raw Data High Register (FRAW_DATAH) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA	Uncorrected Raw Data This register contains the upper 32 bits of the 64-bit raw data used in diagnostic testing of the ECC logic.

4.7.28 Uncorrected Raw Data Low Register (FRAW_DATAL)

NOTE: Raw Data and Raw ECC registers can be loaded with diagnostic values only in diagnostic modes 1 through 6 with DIAG_EN_KEY = 5h in the Diagnostic Control Register (FDIAGCTRL). These modes must be set for at least one clock cycle before writing to any FRAW* register.

Figure 4-35. Uncorrected Raw Data Low Register (FRAW_DATAL) [offset = 74h]

31	0
RAW_DATA[31:0]	
RWP-u	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; -*n* = value after reset -u = Unchanged value on internal reset, cleared on power up

Table 4-40. Uncorrected Raw Data Low Register (FRAW_DATAL) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA	Uncorrected Raw Data. This register contains the lower 32 bits of the 64-bit raw data used in diagnostic testing of the ECC logic.

4.7.29 Uncorrected Raw ECC Register (FRAW_ECC)

NOTE: Raw Data and Raw ECC registers can be loaded with diagnostic values only in diagnostic modes 1 through 6 with DIAG_EN_KEY = 5h in the Diagnostic Control Register (FDIAGCTRL). These modes must be set for at least one clock cycle before writing to any FRAW* register.

Figure 4-36. Uncorrected Raw ECC Register (FRAW_ECC) [offset = 78h]

31								16
Reserved								
R-0								
15					9	8	7	0
Reserved						PIPE_BUF	RAW_ECC	
R-0						R/WC-0	R/WP-u	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; C = Clear by writing a 1; -*n* = value after reset -u = Unchanged value on internal reset, cleared on power up

Table 4-41. Uncorrected Raw ECC Register (FRAW_ECC) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PIPE_BUF		Error came from pipeline buffer hit
		0	This bit is cleared when the RAW_ECC field is updated with new valid information or by writing a 1 to this bit.
		1	Latest error came from a pipeline buffer hit and the FRAW_DATH, FRAW_DATL, and RAW_ECC fields will not contain information that matches the error address nor error status bits.
7-0	RAW_ECC	0-FFh	Uncorrected Raw ECC This register contains the ECC data used in diagnostic testing of the ECC logic.

4.7.30 Parity Override Register (FPAR_OVR)

Figure 4-37. Parity Override Register (FPAR_OVR) [offset = 7Ch]

31	17	16
Reserved		BNK_INV_PAR
R-0		R/WP-0

15	12	11	9	8	7	0
BUS_PAR_DIS		PAR_OVR_KEY		ADD_INV_PAR	DAT_INV_PAR	
R/WP-5h		R/WP-2h		R/WP-0	R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-42. Parity Override Register (FPAR_OVR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	BNK_INV_PAR		Buffer Invert Parity. This bit is only implemented for parity configurations and is reserved for ECC devices.
		0	The SYS_ODD_PARITY value is used.
		1	When 1 and PAR_OVR_KEY = 5h, then the current system parity signal SYS_ODD_PARITY is inverted when doing bank parity calculations. This generates parity errors and causes interrupt signals to be generated.
15-12	BUS_PAR_DIS		Disable Bus Parity. The read data parity is never disabled from this module.
		Ah	The address bus parity error and buffer parity error are disabled and no checking is done and no events are generated.
		All other values	Any other value enables the parity checking on the Address bus and read data bus.
11-9	PAR_OVR_KEY		Parity Override
		5h	The selected ADD_INV_PAR and DAT_INV_PAR fields become active.
		All other values	Any other value causes the module to use the global system parity bit in the system register DEVCR1.
8	ADD_INV_PAR		Address Odd Parity. This bit is active only when PAR_OVR_KEY = 5h. This bit is set to the SYS_ODD_PAR signal value on reset.
		0	The SYS_ODD_PARITY value is used.
		1	The incoming address bus inverts the system signal SYS_ODD_PARITY for parity calculations. This causes parity errors and generate interrupt error signals.
7-0	DAT_INV_PAR		Data Odd Parity. This byte is active only when PAR_OVR_KEY = 5h.
		0	When 0 it will use the SYS_ODD_PARITY value.
		1	The output read data inverts the system signal SYS_ODD_PARITY for parity calculations. This causes parity errors and generates interrupt signals.
			This byte can support up to a 64-bit data bus, but when the device has a 32-bit bus, bits 7:4 are reserved.
			Bit 0 affects read bus bits 7:0, bit 1 affects read bus bits 15:8, and so on. Each active bit of this field is set to the SYS_ODD_PAR signal value on reset.
			The DAT_INV_PAR is used in the parity for the pipeline buffer logic and for the read data bus to the CPU.
			When the ECC logic is in the CPU (CONF_TYPE = 5) and SIL3 is active, this field becomes the ECC corrupting value for SIL3 diagnostic mode 7.
			In diagnostic mode 7, the FPAR_OVR should be set to 00005Axxh to allow writes to the DAT_INV_PAR field. This field should be written before entering diagnostic mode 7.

4.7.31 Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2)

This register is used to disable the SECDED function for one or two sectors from the EEPROM Emulation Flash (bank 7). An additional two sectors can have SECDED disabled by the use of the FEDACSDIS register (see Section 4.7.9).

Figure 4-38. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2) [offset = C0h]

31 29	28	27 24	23 21	20	19 16
BankID3_Inverse	Rsvd	SectorID3_inverse	BankID3	Rsvd	SectorID3
RWP-0	R-0	RWP-0	RWP-0	R-0	RWP-0

15 13	12	11 8	7 5	4	3 0
BankID2_Inverse	Rsvd	SectorID2_inverse	BankID2	Rsvd	SectorID2
RWP-0	R-0	RWP-0	RWP-0	R-0	RWP-0

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-43. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2) Field Descriptions

Bit	Field	Value	Description
31-29	BankID3_Inverse		The bank ID inverse bits are used with the bank ID bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		0	If BankID3 = 7h and BankID3_inverse = 0, then if a valid sector is selected by SectorID3 and SectorID3_inverse that sector will have ECC checking disabled.
		All other values	No sector is disabled by disable ID 3.
28	Reserved	0	Read returns 0. Writes have no effect.
27-24	SectorID3_inverse	0-Fh	The sector ID inverse bits are used with the sector ID bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 3.
23-21	BankID3		The bank ID bits are used with the bank ID inverse bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		7h	If BankID3 = 7h and BankID3_inverse = 0, then if a valid sector is selected by SectorID3 and SectorID3_inverse that sector will have ECC checking disabled.
		All other values	No sector is disabled by disable ID 3.
20	Reserved	0	Read returns 0. Writes have no effect.
19-16	SectorID3	0-Fh	The sector ID bits are used with the sector ID inverse bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 3.
15-13	BankID2_Inverse		The bank ID inverse bits are used with the bank ID bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		0	If BankID2 = 7h and BankID2_inverse = 0, then if a valid sector is selected by SectorID2 and SectorID2_inverse that sector will have ECC checking disabled.
		All other values	No sector is disabled by disable ID 2.
12	Reserved	0	Read returns 0. Writes have no effect.
11-8	SectorID2_inverse	0-Fh	The sector ID inverse bits are used with the sector ID bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 2.
7-5	BankID2		The bank ID bits are used with the bank ID inverse bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		7h	If BankID2 = 7h and BankID2_inverse = 0, then if a valid sector is selected by SectorID2 and SectorID2_inverse that sector will have ECC checking disabled.
		All other values	No sector is disabled by disable ID 2.
4	Reserved	0	Read returns 0. Writes have no effect.

Table 4-43. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2) Field Descriptions (continued)

Bit	Field	Value	Description
3-0	SectorID2	0-Fh	The sector ID bits are used with the sector ID inverse bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 2.

4.7.32 FSM Register Write Enable (FSM_WR_ENA)

Figure 4-39. FSM Register Write Enable (FSM_WR_ENA) [offset = 288h]

31	16
Reserved	
R-0	

15	3	2	0
Reserved		WR_ENA	
R-0		R/WP-2h	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-44. FSM Register Write Enable (FSM_WR_ENA) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
2-0	WR_ENA		FSM Write Enable
		5h	This register must contain 5h in order to write to any other register in the range FFF8 7200h to FFF8 72FFh. This is the first register to be written when setting up the FSM.
		All other values	For all other values, the FSM registers cannot be written.

4.7.33 FSM Sector Register (FSM_SECTOR)

This is a banked register. A separate register is implemented for each bank, but they all occupy the same address. The correct bank must be selected in the FMAC register before reading or writing this register. See Section 4.7.20.

Figure 4-40. FSM Sector Register (FSM_SECTOR) [offset = 2A4h]

31	16
SECT_ERASED[15:0]	
R/WP-1	

15	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-45. FSM Sector Register (FSM_SECTOR) Field Descriptions

Bit	Field	Value	Description
31-16	SECT_ERASED		Sectors Erased
			There is one bit for each sector. Bit 16 corresponds to sector 0, bit 17 corresponds to sector 1, and so on.
		0	During bank erase, each sector whose corresponding bit is 0 will be erased. After bank erase, the bit corresponding to each sector that is erased will be changed from 0 to 1.
		1	During bank erase, each sector whose corresponding bit is 1 will not be erased.
15-0	Reserved	0	These bits are used by the state machine during bank erase. Do not write to these bits.

4.7.34 EEPROM Emulation Configuration Register (EEPROM_CONFIG)

Figure 4-41. EEPROM Emulation Configuration Register (EEPROM_CONFIG) [offset = 2B8h]

31	20	19	16
Reserved		EWAIT	
R-0		R/WP-1h	

15	9	8	7	0
Reserved		AUTOSUSP_EN	AUTOSTART_GRACE	
R-0		R/WP-0	R/WP-2h	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

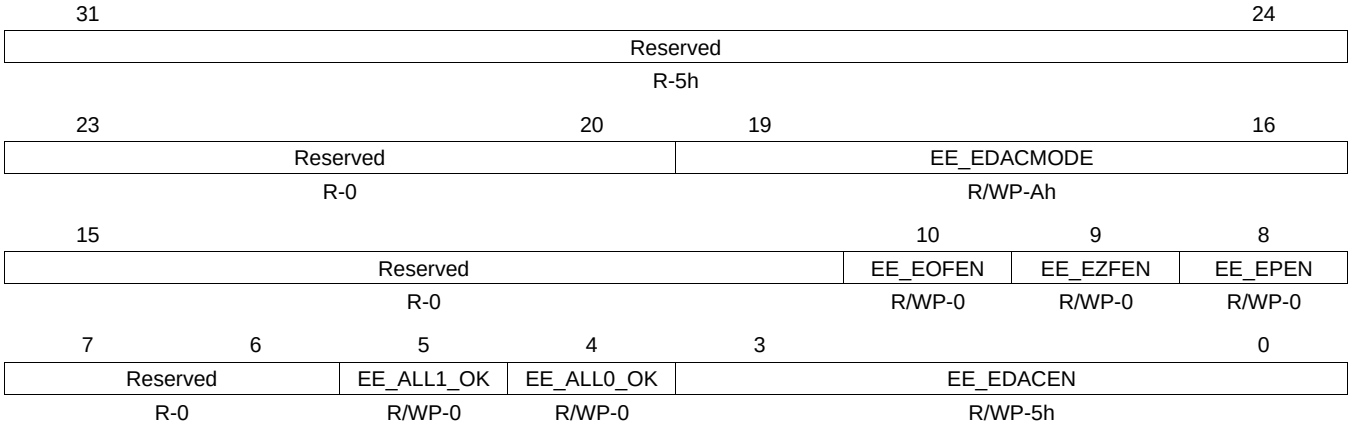
Table 4-46. EPROM Emulation Configuration Register (EEPROM_CONFIG) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	EWAIT	0-Fh	EEPROM Wait state Counter Replaces the RWAIT count in the EEPROM register. The same formulas that apply to RWAIT apply to EWAIT in the EEPROM bank.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	AUTOSUSP_EN		Auto suspend enable The auto-suspend begins when the CPU or Bus2 attempts to access a bank with an active and suspendable FSM operation. If this happens, the FSM automatically issues a suspend command and exits from the FSM. It then does the access. After the access, the FMC waits for a time determined by the Autostart_grace field before issuing the FSM resume command.
		0	Auto suspend is disabled.
		1	Auto suspend is enabled.
7-0	AUTOSTART_GRACE		Auto-suspend Startup Grace Period
		1	The value in this register determines how many cycles the FMC waits after the last CPU or Bus2 access before issuing the FSM resume command.
		0	The FMC waits 16 HCLK periods for each count in the AUTOSTART_GRACE field. A value of 2 waits for 32 periods after the last access. Each access resets the counter to the AUTOSTART_GRACE value × 16.

4.7.35 EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1)

This register controls ECC event detection for the EEPROM Emulation Flash bank (bank 7). For the equivalent register that controls ECC event detection for the main Flash banks, see Section 4.7.2.

Figure 4-42. EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) [offset = 308h]



LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-47. EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	050h	Read returns 050h. Writes have no effect
19-16	EE_EDACMODE		Error Correction Mode for the EEPROM Emulation Flash bank (bank 7). For the main Flash banks, see Section 4.7.2.
		5h	Detection only mode.
			Note: In detection only mode single-bit errors will not be corrected, but will be treated as uncorrectable errors. The single-bit error flags and profiling mode are disabled. Detection only mode has the advantage that a triple bit error will be detected and not mistaken for a single-bit error and mis-corrected.
		All other values	Single-bit errors are corrected and multi-bit or address errors are detected
			Note: It is recommended to leave the EE_EDACMODE field as Ah to guard against soft errors from flipping EE_EDACMODE to a detection only.
15-11	Reserved	0	Reads return 0. Writes have no effect.
10	EE_EOFEN		EEPROM Emulation Event on a correctable One's Fail Enable bit
		0	No ESM event will be generated on a single-bit error when a 1 reads as a 0 and is corrected
		1	An ESM group 1 channel 35 event will be generated on a single-bit error when a 1 reads as a 0 and is corrected
9	EE_EZFEN		EEPROM Emulation Event on a correctable Zero's Fail Enable bit
		0	No ESM event will be generated on a single-bit error when a 0 reads as a 1 and is corrected
		1	An ESM group 1 channel 35 event will be generated on a single-bit error when a 0 reads as a 1 and is corrected
8	EE_EPEN		EEPROM Emulation Error Profiling Enable.
		0	Error profiling is disabled.
		1	Error profiling is enabled.
			An ESM group 1 channel 35 event will be generated when number of correctable bit errors detected and corrected has reached the threshold value defined in the EE_CTRL2 register.
7-6	Reserved	0	Reads return 0. Writes have no effect.

Table 4-47. EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) Field Descriptions (continued)

Bit	Field	Value	Description
5	EE_ALL1_OK		EEPROM Emulation All One Condition Valid.
		0	One condition valid is disabled.
			Reading of an erased location (64 data bits and the corresponding 8 ECC bits are all 1s) will generate ECC errors. The error counter for profiling will increment if all 1s are detected.
		1	One condition valid is enabled.
			Reading of an erased location (64 data bits and the corresponding 8 ECC bits are all 1s) will NOT generate ECC errors. The error counter for profiling will NOT increment if all 1s are detected.
4	EE_ALL0_OK		EEPROM Emulation All Zero Condition Valid.
		0	Zero condition valid is disabled.
			Reading of all 0s (64 data bits and the corresponding 8 ECC bits are all 0s) will generate ECC errors. The error counter for profiling will increment if all 0s are detected.
		1	Zero condition valid is enabled.
			Reading of all 0s (64 data bits and the corresponding 8 ECC bits are all 0s) will NOT generate ECC errors. The error counter for profiling will NOT increment if all 0s are detected.
3-0	EE_EDACEN		EEPROM Emulation Error Detection and Correction Enable
		5h	Error Detection and Correction is disabled
		All other values	Error Detection and Correction is enabled
			Note: It is recommended to leave the EE_EDACEN field as Ah to guard against soft errors from flipping the EE_EDACEN to a disabled state.

4.7.36 EEPROM Emulation Error Correction and Correction Control Register 2 (EE_CTRL2)

Figure 4-43. EEPROM Emulation Error Correction and Correction Control Register 2 (EE_CTRL2) [offset = 30Ch]

31	16
Reserved	
R-0	

15	0
EE_SEC_THRESHOLD	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-48. EEPROM Emulation Error Correction Control Register 2 (EE_CTRL2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	EE_SEC_THRESHOLD	0-FFFFh	EEPROM Emulation Single Error Correction Threshold This register contains the threshold value for the SEC (single error correction) occurrences before a single interrupt request is generated. A threshold of 0 disables the threshold so that it never triggers the profile interrupt.

4.7.37 EEPROM Emulation Correctable Error Count Register (EE_COR_ERR_CNT)

Figure 4-44. EEPROM Emulation Error Corectable Error Count Register (EE_COR_ERR_CNT) [offset = 310h]

31	16
Reserved	
R-0	

15	0
EE_ERRCNT	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP=Write in Privilege Mode; -*n* = value after reset

Table 4-49. EEPROM Emulation Correctable Error Count Register (EE_COR_ERR_CNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	EE_ERRCNT	0-FFFFh	Single Error Correction Count This register contains the number of SEC (single error correction) occurrences. Writing any value to this register resets the count value to 0. The counter resets to 0 when it increments to be equal to the single error correction threshold. This register only increments when profiling mode is enabled. This register is not affected by the EE_ZERO_EN or EE_ONE_EN error control bits in the EE_CTRL1 register.

4.7.38 EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD)

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit to 1 in the FEDACCTRL1 register, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 4-45. EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD) [offset = 314h]

31	16
COR_ERR_ADD	
R-u	

15	3	2	0
COR_ERR_ADD		B_OFF	
R-u		R-u	

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 4-50. EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD) Field Descriptions

Bit	Field	Description
31-3	COR_ERR_ADD	Correctable Error Address COR_ERR_ADD records the CPU logical address of which a correctable error is detected by the ECC logic. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read.
2-0	B_OFF	Byte offset Since ECC is checked on 64 bit data, when checking main memory or OTP, the address captured is aligned to a 64-bit boundary with address bits[2:0] equal to 0. When reading from the ECC bytes, these bits will indicate the failing address of the ECC location associated with the failure. When reading an ECC byte, the ECC is checked against the 64 data bits they protect.

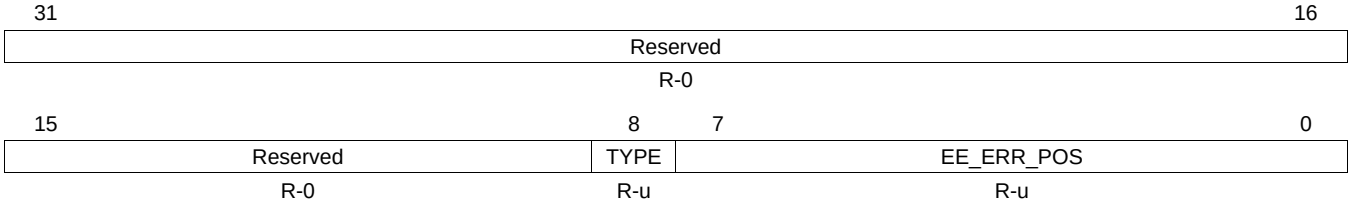
4.7.39 EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS)

The bit position is captured during errors when either EE_EOFEN or EE_EZFEN enable bit is set. During error profiling mode when only EE_EPEN is set, the bit position is not captured if a correctable error is detected. This register is frozen while either the EE_ERR_ZERO_FLG or the EE_ERR_ONE_FLG bit is set in the EE_EDACSTATUS register.

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit to 1 in the FEDACCTRL1 register, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 4-46. EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS) [offset = 318h]



LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 4-51. EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	TYPE		ErrorType
		0	The error was one of the 64 data bits
		1	The error was one of the 8 check bits
7-0	EE_ERR_POS	0-FFh	The bit address of the single-bit error

4.7.40 EEPROM Emulation Error Status Register (EE_STATUS)

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit to 1 in the FEDACCTRL1 register, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

All these error status bits can be deactivated by writing a 1 to the bit; writing a 0 has no effect.

Bits 0 to 3 show correctable errors while bits 4 to 12 show uncorrectable errors. When the uncorrectable errors are triggered, the current address is stored in the EE_UNC_ERR_ADD register.

These error bits are not set while the FMC is in the suspend mode but they can be cleared in suspend by writing 1s to the bits. By setting the SUSP_IGNR bit to 1 in the FEDACCTRL1 register, these error bits can be set in suspend mode.

Figure 4-47. EEPROM Emulation Error Status Register (EE_STATUS) [offset = 31Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved			EE_D_UNC_ERR	Reserved			EE_UNC_ERR
R-0			R/CP-u	R-0			R/CP-u

7	6	5	4	3	2	1	0
Reserved	EE_CMG	Reserved	EE_CME	EE_D_COR_ERR	EE_ERR_ONE_FLG	EE_ERR_ZERO_FLG	EE_ERR_PRF_FLG
R-0	R-0	R-0	R-0	R/CP-u	R/CP-u	R/CP-u	R/CP-u

LEGEND: R = Read only; R/CP=Read and Clear in Privilege Mode; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 4-52. EEPROM Emulation Error Status Register (EE_STATUS) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12	EE_D_UNC_ERR		Diagnostic Mode Uncorrectable Error Status Flag
		0	No uncorrectable error was detected in diagnostic mode 1.
		1	An uncorrectable error was detected in diagnostic mode 1. This means two or more bits in the data or ECC field have been found in error, or one or more bits in the address have been found in error.
11-9	Reserved	0	Reads return 0. Writes have no effect.
8	EE_UNC_ERR		EEPROM Emulation Uncorrectable Error Flag
		0	No uncorrectable errors were detected in bank 7.
		1	An uncorrectable error was detected in bank 7.
7	Reserved	0	Reads return 0. Writes have no effect.
6	EE_CMG		EEPROM Emulation Compare Malfunction Good
		0	Compare malfunction was detected on the Bus2 SECDED logic.
		1	Compare malfunction was not detected on the Bus2 SECDED logic.
5	Reserved	0	Reads return 0. Writes have no effect.
4	EE_CME		EEPROM Emulation Compare Malfunction Error
		0	Compare malfunction was not detected on the Bus2 SECDED logic.
		1	Compare malfunction was detected on the Bus2 SECDED logic.
3	EE_D_COR_ERR		Diagnostic Correctable Error Flag
		0	No correctable error was detected.
		1	A correctable error was detected.

Table 4-52. EEPROM Emulation Error Status Register (EE_STATUS) Field Descriptions (continued)

Bit	Field	Value	Description
2	EE_ERR_ONE_FLG		Error on One Fail Error Flag
		0	No correctable error was detected.
		1	A correctable error was detected.
1	EE_ERR_ZERO_FLG		Error on Zero Fail Error Flag
		0	No correctable error was detected.
		1	A correctable error was detected.
0	EE_ERR_PRF_FLG		Error Profiling Error Flag
		0	No correctable error was detected.
		1	A correctable error was detected.

4.7.41 EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD)

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit to 1 in the FEDACCTRL1 register, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 4-48. EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD) [offset = 320h]

31	16
UNC_ERR_ADD	
R-u	

15	3	2	0
UNC_ERR_ADD		B_OFF	
R-u		R-u	

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 4-53. EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD) Field Descriptions

Bit	Field	Description
31-3	UNC_ERR_ADD	Uncorrectable Error Address UNC_ERR_ADD records the CPU logical address of which an un-correctable error is detected by the ECC logic. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read.
2-0	B_OFF	Byte offset Since ECC is checked on 64 bit data, when checking main memory or OTP, the address captured is aligned to a 64-bit boundary with address bits[2:0] equal to 0. When reading from the ECC bytes, these bits indicate the failing address of the ECC location associated with the failure. When reading an ECC byte, the ECC is checked against the 64 data bits they protect.

4.7.42 Flash Bank Configuration Register (FCFG_BANK)

Figure 4-49. Flash Bank Configuration Register (FCFG_BANK) [offset = 400h]

31	20	19	16
EE_BANK_WIDTH		Reserved	
R-48h		R-1	

15	4	3	0
MAIN_BANK_WIDTH		Reserved	
R-90h		R-2h	

LEGEND: R = Read only; -*n* = value after reset

Table 4-54. Flash Bank Configuration Register (FCFG_BANK) Field Descriptions

Bit	Field	Value	Description
31-20	EE_BANK_WIDTH		Bank 7 width (72 bits wide)
		48h	This read-only value indicates the maximum number of bits that can be programmed in the bank in one operation. The 72 bits includes 64 data bits and 8 ECC bits.
19-16	Reserved	1	Writes have no effect.
15-4	MAIN_BANK_WIDTH		Width of main Flash banks (144 bits wide)
		90h	This read-only value indicates the maximum number of bits that can be programmed in the bank in one operation. The 144 bits includes 128 data bits and 16 ECC bits.
3-0	Reserved	2h	Writes have no effect.

Tightly-Coupled RAM (TCRAM) Module

This chapter describes the tightly-coupled RAM (TCRAM) module.

Topic **Page**

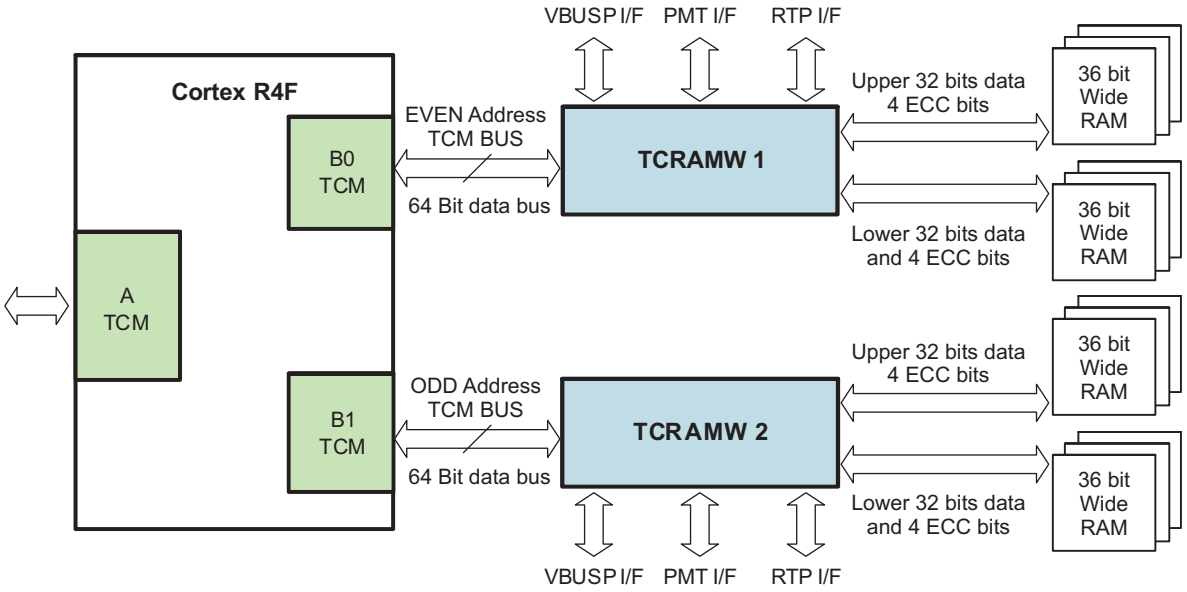
5.1 Overview

The Hercules family of microcontrollers are based on the ARM Cortex-R4 processor. This CPU has two tightly-coupled memory interfaces – ATCM and BTCM, which are used to interface to the program and data memories, respectively. The Hercules MCUs use the ATCM interface for the main flash memory and the BTCM interface for the CPU data RAM.

5.1.1 B0TCM and B1TCM Connection Diagram

The BTCM interface is further divided into two parts – B0TCM and B1TCM, which are both used to interface to actual RAM banks as shown in Figure 5-1.

Figure 5-1. TCRAM Module Connections



5.1.2 Main Features

The main features of the tightly-coupled RAM interface module are:

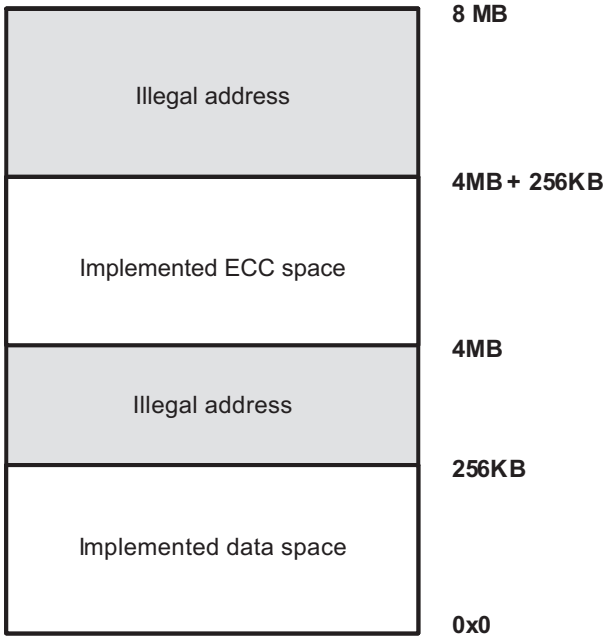
- Controls read/write accesses to the data RAM
- Decodes addresses within the memory region allocated for the RAM
- Supports read and write accesses in 64-bit, 32-bit, 16-bit or 8-bit access sizes
  - Does not support bit-wise operations
- Safety Features:
  - Single-Error-Correction Double-Error-Detection (SECDED)
    - Uses the CPU's Event bus and maintains the SECDED status in memory-mapped registers
    - Captures the number of occurrences of single-bit or multi-bit errors as well as the RAM address that has the fault
    - Generates signals for indicating single-bit and multi-bit errors to the Error Signaling Module (ESM)
  - Parity Protection for BTCM Address Bus and Control Signals
    - Uses the CPU's TCM Address Parity Scheme and indicates an address bus parity error to the ESM

  - Redundant Address Decode Scheme
    - Checks the decoding of CPU address lines and generation of correct memory selects for the RAM banks
- Supports auto-initialization of the CPU data RAM banks

5.2 RAM Memory Map

The ARM Cortex-R4 CPU allows up to 8MB to be accessed through the BTCM interface. The Hercules family of microcontrollers support up to 256KB RAM on the BTCM interface. Check the specific part's datasheet to identify the actual amount of TCRAM supported on the part. This RAM is protected by ECC allowing the CPU to correct any single-bit errors and detect multi-bit errors within a 64-bit value. The error correction codes (ECC) are stored in the RAM memory space as well. The memory map for the TCRAM and the corresponding ECC space is shown in Figure 5-2. Any access to an unimplemented TCRAM location results in an error response from the TCRAM module.

Figure 5-2. RAM Memory Map



Each RAM data word is 64-bits wide. These 64 bits are divided into two 32 bits per RAM bank as shown in Figure 5-1. The 8 bits of ECC are also divided into 4 bits per RAM bank.

For every 64-bit read from the RAM, an 8-bit ECC is also read by the CPU on its ECC bus. Similarly, for every 64-bit write to the RAM, the CPU also writes an 8-bit ECC using the same ECC bus.

NOTE: **Read-Modify-Write Requirement for Writes to RAM:** The TCRAM interface module supports 64-bit, 32-bit, 16-bit, or 8-bit writes to the RAM. However, the ECC is calculated by the CPU for 64-bit values only. For any write access smaller than 64 bits, it is necessary to force the CPU to perform a 64-bit read-modify-write operation in order to ensure that the correct ECC is also written. This can be done by setting the bit 1: BTCMRMW of c15, the Secondary Auxiliary Control Register of the CPU.

The ECC memory can also be directly accessed via memory-mapped offset addresses starting from 4MB, as shown in Figure 5-2. A read from the ECC space results in the 8-bit ECC value appearing on each byte of the 64-bit CPU data. The ECC memory can only be written to as a 64-bit access. The write to the ECC space must also first be enabled via the RAM Control Register (RAMCTRL).

Accesses to the ECC space are not traced out to the RAM Trace Port (RTP).

NOTE: **No ECC Error Generated for Accesses to ECC Memory:** A read from the ECC space sends the ECC value on both the 64-bit TCM read data bus as well as the 8-bit ECC bus. This could result in the detection of a multi-bit error by the SECDED logic inside the CPU. The TCRAM interface module ignores the ECC error indicated by the CPU for the access to the ECC space.

5.3 Safety Features

The TCRAM interface module incorporates some features that are designed specifically with safety considerations. These are described in the following sections.

5.3.1 Support for Cortex-R4 CPU's Single-Error-Correction Double-Error-Detection (SECDED)

The TCRAM interface module monitors the CPU's event bus. The CPU's event bus signals single-bit or multi-bit errors for B0TCM as well as B1TCM separately. These signals are monitored by the TCRAM modules for each of these interfaces.

TCRAM Interface Module Features dedicated for SECDED Support:

- Dedicated single-bit error counter
  - This counter is stored in a memory-mapped register called RAMOCCUR)
  - RAMOCCUR is used to count the single-bit errors corrected by the CPU's SECDED logic
  - The TCRAM interface module allows the application to generate an interrupt via the RAMINTCTRL register when the number of single-bit errors corrected by the CPU exceeds a programmable threshold, RAMTHRESHOLD
- RAM Error Status Register
  - The errors detected by the TCRAM interface module as well as those indicated by the CPU are flagged in the RAMERRSTATUS register
  - There are separate bits to indicate single-bit error, double-bit error, address decode failure, address compare logic failure, read-address parity failure, and write-address parity failure
- ECC Error Address Capture
  - Seperate registers to hold the address on which a single-bit error is detected (RAMSERRADDR) or a double-bit error is detected (RAMUERRADDR)
  - The RAMSERRADDR register is only updated when the RAMTHRESHOLD value is set to 1
  - Both the RAMSERRADDR and RAMUERRADDR capture the 64-bit address for the access to the TCRAM as an offset from the base address of the TCRAM (0x08000000 by default)

NOTE: **Cortex-R4 CPU Event Bus Signaling Not Enabled By Default:** Upon power-up and after a CPU reset, the event signaling mechanism inside the Cortex-R4 CPU is disabled. This feature must be enabled by setting the Export (X-bit) of the Performance Monitoring and Control Register (PMNC) in the CPU. The TCRAM interface module can only capture the single-bit or double-bit ECC error occurrences once the CPU's event signaling mechanism is enabled.

5.3.2 Support for Cortex-R4 CPU's Address and Control Bus Parity Checking

The Cortex-R4 CPU calculates a single parity-bit for the TCRAM address and control signals. The TCRAM interface module also computes this parity bit based on the CPU's address bus and control signals. The computed parity bit is compared against the parity bit received from the CPU. A mismatch is signaled as an Address Parity Failure to the Error Signaling Module (ESM). There is a separate address parity failure error channel for B0TCM and B1TCM.

The 64-bit TCRAM address that fails the parity check is captured in the RAMPERRADDR register as an offset from the base address of the TCRAM (0x08000000 by default). The TCRAM interface module also indicates the type of access, read or write, that failed the parity check. This is indicated by the RADDR_PAR_FAIL or the WADDR_PAR_FAIL status flags in the RAMERRSTATUS register.

The RAMERRSTATUS and RAMPERRADDR registers must be cleared by the application in order for the TCRAM interface module to continue capturing subsequent errors and error addresses.

The parity scheme used for the described parity checking mechanism is defined by the global system parity selection. This can be configured using the DEVPARSEL field of the DEVCR1 control register in the system module. This device-wide parity scheme can be overridden inside the TCRAM interface module by configuring the Address Parity Override field in the RAMCTRL register.

NOTE: **No Change Of Parity Scheme On-The-Fly:** The TCRAM interface module does not support on-the-fly change to the parity scheme being used for checking the CPU address bus and control bus. The application must ensure that the parity polarity (odd or even) is not changed while there is an ongoing access to the TCRAM.

5.3.3 Redundant Address Decode

The TCRAM interface module generates the memory selects for each of the TCRAM banks as well as the ECC memory based on the CPU address. The logic to generate these memory selects is duplicated and the outputs compared to detect any address decode errors. A mismatch is indicated as an Address Error to the Error Signaling Module (ESM), one signal for B0TCM and one for B1TCM. The TCRAM or ECC address that caused the fault is captured in the RAMUERRADDR register. This 64-bit address is stored as an offset from the base of the TCRAM or ECC memory.

As described earlier, each individual physical RAM bank is 36 bits wide. Each RAM bank contributes 32 bits of data and 4 bits of ECC when the bus master performs a 64-bit read from the TCRAM. Each TCRAM bank receives a memory select and the address from the TCRAM interface module. Any difference between the address and the memory selects results in wrong data and ECC pair being sent to the CPU. The CPU's SECDED block will detect this data error.

The TCRAM interface module also supports a mechanism to test the operation of the redundant address decode logic and the compare logic. This testing is supported by providing a test stimulus, and can be triggered by the application by configuring the RAMTEST register. The address of any error identified during testing of the redundant address decode and compare logic is not captured in the RAMUERRADDR register.

NOTE: **Address decode checking when in compare logic test mode:** When the address decode and compare logic test mode is enabled, the redundant address decode and compare logic is not available for checking the proper generation of the memory selects for the TCRAM and ECC memory.

5.4 TCRAM Auto-Initialization

The RAM memory can be initialized by using the dedicated auto-initialization hardware. The TCRAM Module initializes the entire memory when the auto-init is enabled for the RAM. All RAM data memory is initialized to zeros and the ECC memory is initialized to the correct ECC value for zeros, that is, 0Ch.

5.5 Emulation / Debug Mode Behavior

The following describes the behavior of the TCRAM Module when in debug mode:

- The RAMOCCUR register continues to count the single-bit error corrections performed by the Cortex-R4 CPU's SECDED logic.
- No single-bit error interrupt is generated nor is any single-bit error address captured even when the RAMOCCUR counter reaches the programmed single-bit error correction threshold.
- No uncorrectable error interrupt is generated nor is any double-bit error address captured.
- No address parity error interrupt is generated nor is any parity error address captured.
- The RAMUERRADDR register is not cleared by a read in debug mode.
  - That is, if a double-bit error address is captured and is not read by the CPU before entering debug mode, then it remains frozen during debug mode even if it is read.
- The RAMPERRADDR register is not cleared by a read in debug mode.

5.6 TCRAM Control and Status Registers

The TCRAM Module registers listed in Table 5-1 are accessed through the system module register space in the Cortex-R4 CPU's memory map. All registers are 32-bit wide and are located on a 32-bit boundary. Reads and writes to registers are supported in 8-, 16-, and 32-bit accesses. The base address for the control registers is FFFF F800h for even RAM ECC and FFFF F900h for odd RAM ECC.

Table 5-1. TCRAM Module Control and Status Registers

Offset	Acronym	Register Description	Section
00h	RAMCTRL	TCRAM Module Control Register	Section 5.6.1
04h	RAMTHRESHOLD	TCRAM Module Single-Bit Error Correction Threshold Register	Section 5.6.2
08h	RAMOCCUR	TCRAM Module Single-Bit Error Occurrences Control Register	Section 5.6.3
0Ch	RAMINTCTRL	TCRAM Module Interrupt Control Register	Section 5.6.4
10h	RAMERRSTATUS	TCRAM Module Error Status Register	Section 5.6.5
14h	RAMSERRADDR	TCRAM Module Single-Bit Error Address Register	Section 5.6.6
1Ch	RAMUERRADDR	TCRAM Module Uncorrectable Error Address Register	Section 5.6.7
30h	RAMTEST	TCRAM Module Test Mode Control Register	Section 5.6.8
38h	RAMADDRDECVECT	TCRAM Module Test Mode Vector Register	Section 5.6.9
3Ch	RAMPERADDR	TCRAM Module Parity Error Address Register	Section 5.6.10

5.6.1 TCRAM Module Control Register (RAMCTRL)

The RAMCTRL register, shown in Figure 5-3 and described in Table 5-2, controls the safety features supported by the TCRAM Module.

Figure 5-3. TCRAM Module Control Register (RAMCTRL) (offset = 00h)

31	30	29	28	27	24
Reserved	EMU_TRACE_DIS	Reserved		ADDR_PARITY_OVERRIDE	
R-0	R/WP-0	R-0		R/WP-0	

23	20	19	16
Reserved		ADDR_PARITY_DISABLE	
R-0		R/WP-5h	

15	9	8
Reserved		ECC_WR_EN
R-0		R/WP-0

7	4	3	0
Reserved		ECC_DETECT_EN	
R-0		R/WP-Ah	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 5-2. TCRAM Module Control Register (RAMCTRL) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reads return 0. Writes have no effect.
30	EMU_TRACE_DIS		Emulation Mode Trace Disable. This bit, when set, disables the tracing of read data to RAM Trace Port (RTP) module during emulation mode access.
		0	Data is allowed to be traced out to the trace modules during emulation mode accesses.
		1	Data is blocked from being traced out to the trace modules during emulation mode accesses.
29-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	ADDR_PARITY_OVERRIDE		Address Parity Override. This field, when set to Dh, will invert the parity scheme selected by the device global parity selection. The address parity checker would then work on the inverted parity scheme. By default, the parity scheme is the same as the global device parity scheme.
		Dh	Parity scheme is opposite to the device global parity scheme.
		All other values	Parity scheme is the same as the device global parity scheme.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	ADDR_PARITY_DISABLE		Address Parity Detect Disable. This field, when set to Ah, disables the parity checking for the address bus. The parity checking is enabled when this field is set to any other value. **Note: The application must ensure that the WADDR_PAR_FAIL and RADDR_PAR_FAIL bits in RAMERRSTATUS register are cleared before enabling address parity checking.**
		Ah	Address parity checking is disabled.
		All other values	Address parity checking is enabled.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	ECC_WR_EN		ECC Memory Write Enable. This bit is provided to prevent accidental writes to the ECC memory. A write access to the ECC memory is allowed only when the ECC_WR_EN bit is set to 1. If this bit is cleared, then any writes to ECC memory are ignored. **Note: Reads are allowed from the ECC memory regardless of the state of the ECC_WR_EN bit.**
		0	ECC memory writes are disabled.
		1	ECC memory writes are enabled.

Table 5-2. TCRAM Module Control Register (RAMCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	ECC_DETECT_EN		ECC Detect Enable. This is a 4-bit key to enable the ECC detection feature in the TCRAM Module. If this field is set to any value other than 5h, then the TCRAM Module starts monitoring the TCM event bus and generates the corresponding error status flags. The error status updates are done only when the ECC_DETECT_EN field is not 5h. The ECC detection is enabled by default as the ECC_DETECT_EN default value is Ah.
		5h	ECC detection is disabled.
		All other values	ECC detection is enabled.

5.6.2 TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD)

The RAMTHRESHOLD register, shown in Figure 5-4 and described in Table 5-3, allows the application to configure the number of single-bit error corrections by the SECDED logic inside the Cortex-R4 CPU before generating a single-bit error interrupt.

Figure 5-4. TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD) (offset = 04h)

31	16
Reserved	
R-0	

15	0
THRESHOLD	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

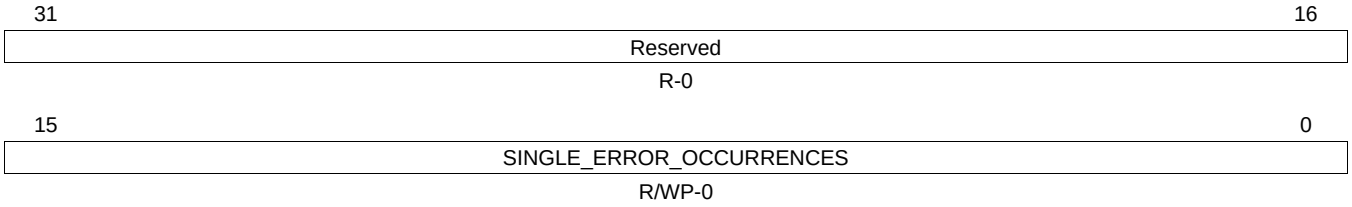
Table 5-3. TCRAM Module Single-Bit Error Correction Threshold Register (RAMTHRESHOLD) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	THRESHOLD	0-FFFFh	Single-bit Error Threshold Count. This field contains the threshold value for the Single-bit Error Correction (SEC) occurrences before the single-bit error interrupt is generated. If this threshold is set to 1 then all single-bit error addresses are captured. To enable the error occurrence detection, the threshold must be set to a non-zero value.

5.6.3 TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR)

The RAMOCCUR register, shown in Figure 5-5 and described in Table 5-4, indicates the number of single-bit error corrections performed by the SEC logic inside the Cortex-R4 CPU.

Figure 5-5. TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR) (offset = 08h)



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

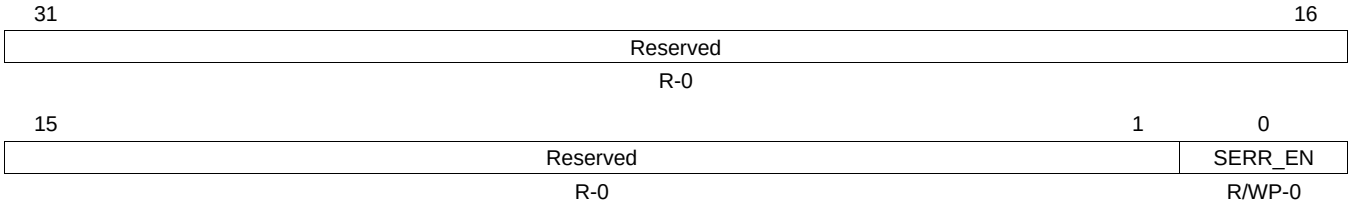
Table 5-4. TCRAM Module Single-Bit Error Occurrences Counter Register (RAMOCCUR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	SINGLE_ERROR_OCCURRENCES	0-FFFFh	Single-bit Error Correction Occurrences. This 16-bit counter contains the number of single-bit error occurrences. RAMOCCUR is reset to zero when it becomes equal to the THRESHOLD value set in the RAMTHRESHOLD register. The application must clear the RAMOCCUR register by writing 0 before setting the THRESHOLD value. If the RAMOCCUR value is already higher than the programmed THRESHOLD value then the counter increments and wraps around (overflow) to zero. **Note: If the application tries to clear the RAMOCCUR register at the same time as the TCRAM Module tried to update it, then the TCRAM Module takes priority.** **Note: When the RAMTHRESHOLD register is set to 1, then the RAMOCCUR register must be cleared whenever a single-bit error correction occurs in order to count subsequent single-bit error corrections.**

5.6.4 TCRAM Module Interrupt Control Register (RAMINTCTRL)

The RAMINTCTRL register, shown in Figure 5-6 and described in Table 5-5, enables the generation of an interrupt to the CPU whenever the number of single-bit error corrections (RAMOCCUR) reaches the programmed threshold (RAMTHRESHOLD).

Figure 5-6. TCRAM Module Interrupt Control Register (RAMINTCTRL) (offset = 0Ch)



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 5-5. TCRAM Module Interrupt Control Register (RAMINTCTRL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	SERR_EN		Single-bit Error Correction Interrupt Enable. This bit, when set to 1, enables the generation of the single-bit error interrupt when the RAMOCCUR count reaches the programmed RAMTHRESHOLD. If the interrupt is not enabled, the single-bit error counter continues to count by resetting back to zero without generating any error interrupt. The SERR status flag in the RAMERRSTATUS register gets set regardless of whether the SERR interrupt is enabled or not.
		0	Single-bit error generation is disabled.
		1	Single-bit error generation is enabled.

5.6.5 TCRAM Module Error Status Register (RAMERRSTATUS)

The RAMERRSTATUS register, shown in Figure 5-7 and described in Table 5-6, indicates the status of the various error conditions monitored by the TCRAM Module.

Figure 5-7. TCRAM Module Error Status Register (RAMERRSTATUS) (offset = 10h)

31	16
Reserved	
R-0	

15	10	9	8
Reserved		WADDR_PAR_FAIL	RADDR_PAR_FAIL
R-0		R/W1CP-0	R/W1CP-0

7	6	5	4	3	2	1	0
Reserved		DERR	ADDR_COMP_LOGIC_FAIL	Reserved	ADDR_DEC_FAIL	Reserved	SERR
R-0		R/W1CP-0	R/W1CP-0	R-0	R/W1CP-0	R-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 5-6. TCRAM Module Error Status Register (RAMERRSTATUS) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	WADDR_PAR_FAIL	0-1	This bit indicates a Write Address Parity Failure. This bit must be cleared by writing 1 to it in order to enable the capture of parity error address for subsequent failures. This bit must be in a cleared state for generation of any new parity error interrupt.
8	RADDR_PAR_FAIL	0-1	This bit indicates a Read Address Parity Failure. This bit must be cleared by writing 1 to it in order to enable the capture of parity error address for subsequent failures. This bit must be in a cleared state for generation of any new parity error interrupt.
7-6	Reserved	0	Reads return 0. Writes have no effect.
5	DERR	0-1	This bit indicates a multi-bit error detected by the Cortex-R4 SECDED logic.
4	ADDR_COMP_LOGIC_FAIL	0-1	Address decode logic element failed. This bit indicates that the redundant address decode logic test scheme has detected that a compare element has malfunctioned during the testing of the logic. This bit has to be cleared by writing 1 to it in order to enable the capture of uncorrectable error address for subsequent failures. This bit has to be in a cleared state for generation of a new uncorrectable error interrupt. This bit only gets set in the test mode, and has no relevance in functional mode.
3	Reserved	0	Reads return 0. Writes have no effect.
2	ADDR_DEC_FAIL	0-1	Address decode failed. This bit indicates that an address error interrupt was generated by the redundant address decode and compare logic due to a functional failure. This bit must be cleared by writing 1 to it in order to enable the capture of uncorrectable error address for subsequent failures. This bit has to be in a cleared state for generation of a new address error interrupt.
1	Reserved	0	Reads return 0. Writes have no effect.
0	SERR	0-1	Single Error Status. This bit indicates that the single-bit error threshold has been reached. This bit is set even if the single-bit error threshold interrupt is disabled. This bit must be cleared by writing 1 to it in order to clear the interrupt request and to enable subsequent single-bit error interrupt generation.

5.6.6 TCRAM Module Single-Bit Error Address Register (RAMSERRADDR)

The RAMSERRADDR register, shown in Figure 5-8 and described in Table 5-7, captures the address for which the Cortex-R4 CPU detected a single-bit error.

NOTE: The SERR bit in the RAMERRSTATUS register must be cleared, by writing 1 to the bit, in order to enable the RAMSERRADDR register to capture a subsequent new error address.

Figure 5-8. TCRAM Module Single-Bit Error Address Register (RAMSERRADDR) (offset = 14h)

31	18	17	16
Reserved		SINGLE_ERROR _ ADDRESS	
R-0		R-0	

15	3	2	0
SINGLE_ERROR_ADDRESS		Reserved	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-7. TCRAM Module Single-Bit Error Address Register (RAMSERRADDR) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reads return 0. Writes have no effect.
17-3	SINGLE_ERROR_ADDRESS	0-7FFFh	This register captures the bits 17-3 of the address for which the Cortex-R4 CPU detects a single-bit error when the RAMTHRESHOLD register is set to 1. The lower 3 bits are always tied to zero so that the address captured is a double-word (64-bit) address. This is a 64-bit address is stored as an offset from the base of the TCRAM or ECC memory. This register can only be reset by asserting power-on reset, and holds the last error address even after a system reset.
2-0	Reserved	0	Reads return 0. Writes have no effect.

5.6.7 TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR)

The RAMUERRADDR register, shown in Figure 5-9 and described in Table 5-8, captures the address for which the Cortex-R4 CPU detected a multi-bit error.

Figure 5-9. TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR) (offset = 1Ch)

31	23	22	16
Reserved		UNCORRECTABLE_ERROR_ADDRESS	
R-0		R-0	

15	3	2	0
UNCORRECTABLE_ERROR_ADDRESS		Reserved	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 5-8. TCRAM Module Uncorrectable Error Address Register (RAMUERRADDR) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reads return 0. Writes have no effect.
22-3	UNCORRECTABLE_ERROR_ADDRESS	0-FFFFFh	This register captures the address for which there was an uncorrectable error. The uncorrectable error is indicated by the Cortex-R4 CPU's SECDED logic and the address error is indicated by the TCRAM Module's redundant address decode and compare logic. For the SECDED multi-bit or double-bit uncorrectable error, this register stores bits 17-3 of the TCM access address. The lower 3 bits 2-0 are always read as zeros to indicate that the latched address is a double-word address. The address bits 31-18 are read as zeros. This is a 64-bit address stored as an offset from the base of the TCRAM or ECC memory. For a redundant address decode and compare logic error, this register stores the complete TCM access address rounded to a double-word boundary (bits 22-3). This error is also indicated by the ADDR_DEC_FAIL status flag in the RAMERRSTATUS register. The register has to be read-cleared to enable further error address captures. Reading the register does not clear its contents but enables the register to be updated with an uncorrectable error address. This register can only be reset by asserting power-on reset, and holds the last error address even after a system reset.
2-0	Reserved	0	Reads return 0. Writes have no effect.

5.6.8 TCRAM Module Test Mode Control Register (RAMTEST)

The RAMTEST register, shown in Figure 5-10 and described in Table 5-9, controls the test mode of the TCRAM Module.

Figure 5-10. TCRAM Module Test Mode Control Register (RAMTEST) (offset = 30h)

31	16
Reserved	
R-0	

15	9	8	7	6	5	4	3	0
Reserved		TRIGGER	TEST_MODE		Reserved		TEST_ENABLE	
R-0		R/WP-0	R/WP-0		R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 5-9. TCRAM Module Test Mode Control Register (RAMTEST) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	TRIGGER	0-1	Test Trigger. This is an auto reset test trigger used to test the redundant address decode and compare logic. A redundant address decode test is executed when test mode is enabled and the test trigger is applied by writing a 1 to this bit. The trigger is valid only if test is enabled and the correct mode is configured in the TEST_MODE field, and the ADDR_DEC_FAIL, ADDR_COMP_LOGIC_FAIL, and DERR flags are cleared in the RAMERRSTATUS register and the RAMUERRADDR register is read-cleared.
7-6	TEST_MODE	0-3h	Test Mode. This field selects either equality of inequality testing schemes. If TEST MODE is set to 2h, equality check is done. The test stimulus stored in ADDRTEST_VECT register is fed directly to both the channels of the comparator. If the XOR of these two inputs **is not zero**, then UERR interrupt is generated and ADDR_COMP_LOGIC_FAIL flag is set in RAMERRSTATUS register. If TEST MODE is set to 1h, inequality check is done. The test stimulus stored in ADDRTEST_VECT register is inverted and fed into one channel and the non-inverted vector is fed into the other channel. If the XOR of these inputs **is zero**, then the UERR interrupt is generated and ADDR_COMP_LOGIC_FAIL flag is set in RAMERRSTATUS register.
5-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	TEST_ENABLE		Test Enable. This is a 4-bit key to enable the redundant address decode and compare logic test scheme. If the test scheme is enabled then the compare logic uses the test vector inputs from the ADDRTEST_VECT register. The functional path comparison is disabled when test mode is enabled.
		Ah	Test mode is enabled.
		All other values	Test mode is disabled.

5.6.9 TCRAM Module Test Mode Vector Register (RAMADDRDECVECT)

The RAMADDRDECVECT register, shown in Figure 5-11 and described in Table 5-10, is used for testing the redundant address decode and compare logic of the TCRAM Module.

Figure 5-11. TCRAM Module Test Mode Vector Register (RAMADDRDECVECT) (offset = 38h)

31	27	26	25	16
Reserved		ECC_SELECT	Reserved	
R-0		R/WP-0	R-0	

15	0
RAM_CHIP_SELECT	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 5-10. TCRAM Module Test Mode Vector Register (RAMADDRDEVECT) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26	ECC_SELECT	0-1	ECC Select. This bit is used to store the ECC select value for the redundant address decode and compare logic. The stored value is passed as test stimulus for the built-in test scheme.
25-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	RAM_CHIP_SELECT	0-FFFFh	RAM Chip Select. This field is used to store the RAM chip select value for the redundant address decode and compare logic. The stored value is passed as test stimulus for the built-in test scheme.

5.6.10 TCRAM Module Parity Error Address Register (RAMPERRADDR)

The RAMPERRADDR register, shown in Figure 5-12 and described in Table 5-11, stores the address for which an address-parity error was detected.

Figure 5-12. TCRAM Module Parity Error Address Register (RAMPERRADDR) (offset = 3Ch)

31	23	22	16
Reserved		ADDRESS_PARITY_ERROR_ADDRESS	
R-0		R-u	

15	3	2	0
ADDRESS_PARITY_ERROR_ADDRESS		Reserved	
R-u		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-11. TCRAM Module Parity Error Address Register (RAMPERRADDR) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reads return 0. Writes have no effect.
22-3	ADDRESS_PARITY_ERROR_ADDRESS	0-FFFFFh	Parity Error Address. This register stores the double-word boundary (bits 22-3) of the TCM access address for which there was an address parity error. This register must be read-cleared to enable further error address captures. Reading the register does not clear the register contents but enables the register to be updated with a new parity error address. This is a 64-bit address is stored as an offset from the base of the TCRAM or ECC memory.
2-0	Reserved	0	Reads return 0. Writes have no effect.

Programmable Built-In Self-Test (PBIST) Module

This chapter describes the programmable built-in self-test (PBIST) controller module used for testing the on-chip memories.

Topic **Page**

6.1 Overview

The PBIST (Programmable Built-In Self-Test) controller architecture provides a run-time-programmable memory BIST engine for varying levels of coverage across many embedded memory instances.

6.1.1 Features of PBIST

- Information regarding on-chip memories, memory groupings, memory background patterns and test algorithms stored in dedicated on-chip PBIST ROM
- Host processor interface to configure and start BIST of memories
- Supports testing of PBIST ROM itself as well
- Supports testing of each memory at its maximum access speed in application
- Implements intelligent clock gating to conserve power
- Execution of microcode from PBIST ROM supported for ROM clock speeds up to 112 MHz. Reference the device specific datasheet for maximum PBIST execution frequency.

6.1.2 PBIST versus Application Software-Based Testing

The PBIST architecture consists of a small coprocessor with a dedicated instruction set targeted specifically toward testing memories. This coprocessor executes test routines stored in the PBIST ROM and runs them on multiple on-chip memory instances. The on-chip memory configuration information is also stored in the PBIST ROM.

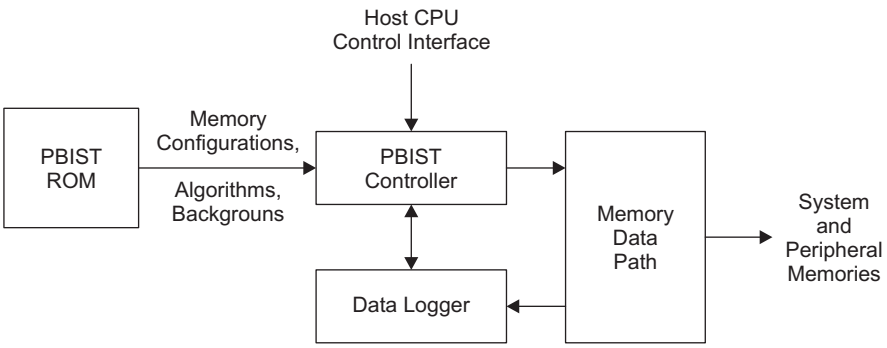
The PBIST Controller architecture offers significant advantages over tests running on the main Cortex-R4 processor (application software-based testing):

- Embedded CPUs have a long access path to memories outside the tightly-couple memory sub-system, while the PBIST controller has a dedicated path to the memories specifically for the self-test.
- Embedded CPUs are designed for their targeted use and are often not easily programmed for memory test algorithms.
- The memory test algorithm code on embedded CPUs is typically significantly larger than that needed for PBIST.
- The embedded CPU is significantly larger than the PBIST controller.

6.1.3 PBIST Block Diagram

Figure 6-1 illustrates the basic PBIST blocks and its wrapper logic for the device.

Figure 6-1. PBIST Block Diagram



6.1.3.1 On-chip ROM

The on-chip ROM contains the information regarding the algorithms and memories to be tested.

6.1.3.2 Host Processor Interface to the PBIST Controller Registers

The Cortex-R4 CPU can select the algorithm and RAM groups for the memories' self-test from the on-chip ROM based on the application requirements. Once the self-test has executed, the CPU can query the PBIST controller registers to identify any memories that failed the self-test and to then take appropriate next steps as required by the application's author.

6.1.3.3 Memory Data Path

This is the read and write data path logic between different system and peripheral memories tightly coupled to the PBIST memory interface. The PBIST controller executes each selected algorithm on each valid memory group sequentially until all the algorithms are executed.

NOTE: Not all algorithms are designed to run on all RAM groups. If an algorithm is selected to run on an incompatible memory, this will result in a failure. Refer to Table 2-5 and Table 2-6 for RAM grouping and algorithm information.

6.2 RAM Grouping and Algorithm

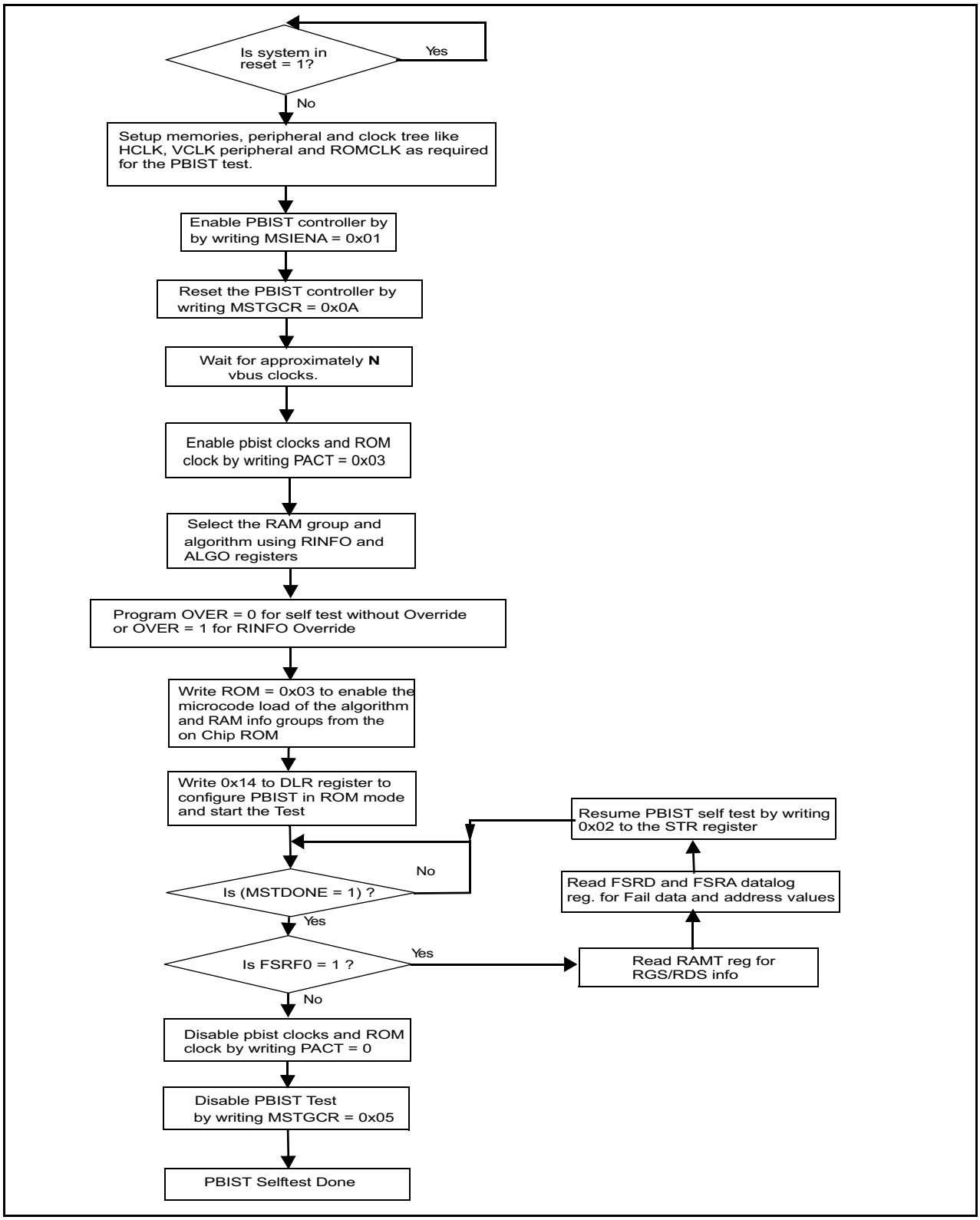
Table 2-5 gives the list of RAM groups and their types supported on the device. Table 2-6 maps the different algorithms supported in application mode for the RAM groups with the background patterns used for the particular algorithm.

NOTE: March13 is the most recommended algorithm for the memory self-test.

6.3 PBIST Flow

Figure 6-1 illustrates the memory self-test flow.

Figure 6-2. PBIST Memory Self-Test Flow Diagram



6.3.1 PBIST Sequence

1. Configure the device clock sources and domains so that they are running at their target frequencies.
2. Program the HCLK to PBIST ROM clock ratio by configuring the ROM_DIV field (bits 9:8) of the MSTGCR register in the system module. This device supports a maximum PBIST ROM clock frequency of HCLK.
3. Enable PBIST Controller by setting bit 0 of MSIENA register in the system module.
4. Enable the PBIST self-test by writing a value of Ah to bits 3:0 of the MSTGCR in the system module.
5. Wait for N VBUS clock cycles based on the HCLK to PBIST ROM clock ratio:

   N = 16 when HCLK:PBIST ROM clock is 1:1
   N = 32 when HCLK:PBIST ROM clock is 1:2
   N = 64 when HCLK:PBIST ROM clock is 1:4
   N = 64 when HCLK:PBIST ROM clock is 1:8
6. Write 1h to PACT register to enable the PBIST internal clocks.
7. Program the ALGO register to decide which algorithm from the instruction ROM must be selected (the default value of ALGO register is all 1s, meaning all algorithms are selected). Similarly, program the RINFOL and RINFOU registers to indicate whether a particular RAM group in the instruction ROM would get executed or not.

   NOTE: In case of RAM Override (Override Register (OVER) = 00), the user should make sure that only the algorithms that run on similar RAMs are selected. If a single port algorithm is selected in ROM Algorithm Mask Register (ALGO), the RAM Info Mask Lower Register (RINFOL) and RAM Info Mask Upper Register (RINFOU) must select only the single port RAM's. The same applies for two port RAM's. Check Table 2-5 for information on the memory types.
8. Program OVER = 1h to run PBIST self-test without RAM override. Program OVER = 0 to run PBIST self-test with RAM Override.
9. Write a value of 3h to the ROM mask register should the microcode for the Algorithms as well as the RAM groups loaded from the on chip PBIST ROM.
10. Write DLR (Data Logger register) with 14h to configure the PBIST run in ROM mode and to enable the configuration access. This starts the memory self-tests.
11. Wait for the PBIST self-test done by polling MSTDONE bit of MSTCGSTAT register in the system module.
12. Once self-test is completed, check the Fail Status register FSRF0.

    In case there is a failure (FSRF0 = 1h):
    a. Read RAMT register which indicates the RGS and RDS values of the failure RAM.
    b. Read FSRC0 and FSRC1 registers which contains the failure count.
    c. Read FSRA0 and FSRA1 registers which contains the address of first failure.
    d. Read FSRDL0 and FSRDL1 registers which contains the failure data.
    e. Write a value of 2h to the STR register to resume the test.

    In case there is no failure (FSRF0 = 0), the memory self-test is completed.
    a. Disable the PBIST internal clocks by writing a 0 to the PACT register.
    b. Disable the PBIST self-test by writing a value of 5h to bits 3:0 of the MSTGCR in the system module.

    Repeat steps 2 through 9 for subsequent runs with different RAM group and algorithm configurations.
13. After required Memory tests are completed, Resume or Start the Normal Application software.

NOTE: The contents of the selected memory before the test will be completely lost. User software must take care of data backup if required. Typically the PBIST tests are carried out at the beginning of Application software.

Memory test fail information is reported in terms of RGS:RDS and not RAM GROUP. Check Table 2-5 for information on the RGS:RDS information applicable to each memory being tested.

6.4 Memory Test Algorithms on the On-chip ROM

This section provides a brief description for some of the test algorithms used for memory self-test.

1. **March13N:**
   - March13N is the baseline test algorithm for SRAM testing. It provides the highest overall coverage. The other algorithms provide additional coverage of otherwise missed boundary conditions of the SRAM operation.
   - The concept behind the general march algorithm is to indicate:
     – The bit cell can be written and read as both a 1 and a 0.
     – The bits around the bit cell do not affect the bit cell.
   - The basic operation of the march is to initialize the array to a know pattern, then march a different pattern through the memory.
   - Type of faults detected by this algorithm:
     – Address decoder faults
     – Stuck-At faults
     – Coupled faults
     – State coupling faults
     – Parametric faults
     – Write recovery faults
     – Read/write logic faults
2. **Map Column:**
   - The MAP COLUMN algorithm is used to identify bit line sensitivities in the memory array. The memory array is loaded with a row stripe pattern of all 1s in the first row followed by all 0s in the second row and repeated throughout the array. Then the values are read down each column on consecutive cycles. The pattern in memory is inverted and run the column reads again.
   - This particular pattern is looking for the following SRAM failure mechanisms:
     – Leakage due to a low resist path in a bit
     – An Open in the bit cell
     – Leakage on a BIT or BITN line
     – Miss-balance in the sense amp
     – Leakage in the sense
     – High resist in the sense amp
     – Failure of the pre-charge circuits after read operations

3. **Pre-Charge:**
   - The Pre-Charge algorithm exercises the pre-charge capability within the SRAM array. It is important to specifically target this issue as it is the only part of the analog portion of the SRAM that is frequency sensitive.
   - Similar to the MAP COLUMN algorithm, this algorithm works its way down the columns of the SRAM. However, unlike the MAP COLUMN, this algorithm sandwiches a write between two reads to force the worst-case conditions for the pre-charge circuits in the array.
   - This test will fail when an increase in system frequency nears the minimum access time of the array, at this boundary:
     – High voltage should operate better than low voltage.
     – Likewise, low temperature should operate better than high temperature.
   - If devices fail this test within the operational range of the CPU, then this is a very good test for characterizing the CPU/memory system.
4. **DOWN1a:**
   - The Down1 pattern forces the switching of all data bits and most address bits on consecutive read cycles. This is primarily a read/write test of the CPU/memory subsystem.
   - The aggressive writes target at-speed write failures.
   - It also targets row/column decode in the memory array.
   - Targets the sense amps and sense amp multiplexors.
   - Memory array output buffers.
   - This algorithm operates as follows:
     – Load 1st half of the memory under test with one pattern.
     – Load 2nd half of the memory under test with the bit-wise inverse of the pattern.
     – Alternate sequential reads sequences between one sequence starting at the first of the array and a second sequence starting at the end of the array.
     – Upon completion of the read back, invert the patterns in both halves of the array and repeat the above step.
     – Perform an aggressive write sequence by alternate write between the bottom half of the memory upwards with an data pattern and the top half of the memory downwards with the inverse data pattern.
     – Invert the data pattern for the above two steps to performing another sequence of aggressive writes.
5. **DTXN2a:**

This algorithm is used to target the global column decode Logic.

6.5 PBIST Control Registers

PBIST controller uses configuration registers for programming the algorithm and its execution. All the configuration registers are memory mapped for access by the CPU through the Peripheral Bus interface. The base address for the control registers is FFFF E400h.

NOTE: There is no watchdog functionality implemented in the PBIST controller. If a bad code is executed, the PBIST runs forever. The PBIST controller does not guard against this situation.

Registers are accessible only when the clock to the PBIST controller is active. The clock is activated by first writing 1h to the PACT register.

Table 6-1. PBIST Registers

Offset	Acronym	Register Description	Section
160h	RAMT	RAM Configuration Register	Section 6.5.1
164h	DLR	Datalogger Register	Section 6.5.2
16Ch	PCR	Program Control Register	Section 6.5.3
180h	PACT	PBIST Activate/Clock Enable Register	Section 6.5.4
184h	PBISTID	PBIST ID Register	Section 6.5.5
188h	OVER	Override Register	Section 6.5.6
190h	FSRF0	Fail Status Fail Register 0	Section 6.5.7
198h	FSRC0	Fail Status Count Register 0	Section 6.5.8
19Ch	FSRC1	Fail Status Count Register 1	Section 6.5.8
1A0h	FSRA0	Fail Status Address 0 Register	Section 6.5.9
1A4h	FSRA1	Fail Status Address 1 Register	Section 6.5.9
1A8h	FSRDL0	Fail Status Data Register 0	Section 6.5.10
1B0h	FSRDL1	Fail Status Data Register 1	Section 6.5.10
1C0h	ROM	ROM Mask Register	Section 6.5.11
1C4h	ALGO	ROM Algorithm Mask Register	Section 6.5.12
1C8h	RINFOL	RAM Info Mask Lower Register	Section 6.5.13
1CCh	RINFOU	RAM Info Mask Upper Register	Section 6.5.14

6.5.1 RAM Configuration Register (RAMT)

This register is divided into following internal registers, none of which have a default value after reset. Figure 6-3 and Table 6-2 illustrate this register.

This register provides the information regarding the memory being currently tested. In case of a PBIST failure, the application can read this register to identify the RGS:RDS values for the memory that failed the self-test.

Figure 6-3. RAM Configuration Register (RAMT) (offset = 160h)

31	24	23	16
RGS		RDS	
R/W-x		R/W-x	

15	8	7	6	5	2	1	0
DWR		SMS		PLS		RLS	
R/W-x		R/W-x		R/W-x		R/W-x	

LEGEND: R/W = Read/Write; R = Read only; -x = value is unknown after reset

Table 6-2. RAM Configuration Register (RAMT) Field Descriptions

Bit	Field	Description
31-24	RGS	Ram Group Select. Refer to Table 2-5 for information on the RGS value for each memory.
23-16	RDS	Return Data Select. Refer to Table 2-5 for information on the RDS values for each memory.
15-8	DWR	Data Width Register
7-6	SMS	Sense Margin Select Register
5-2	PLS	Pipeline Latency Select
1-0	RLS	RAM Latency Select

6.5.2 Datalogger Register (DLR)

This register puts the PBIST controller into the appropriate comparison modes for data logging. Figure 6-4 and Table 6-3 illustrate this register.

- **DLR2: ROM-based testing mode**

Writing a 1 to this register starts the ROM-based testing. This register is used to initiate ROM-based testing from Config and ATE interfaces. Also, since a 1 in this bit position means the instruction ROM is used for memory testing, all the intermediate interrupts and PBIST done signal after each memory test are masked until all the selected algorithms in the ROM are executed for all RAM groups. However, a failure would stop the test and report the status immediately.

- **DLR4: Configuration access mode**

This mode, when set, indicates the CPU is being used to access PBIST.

Figure 6-4. Datalogger Register (DLR) (offset = 164h)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		DLR4	Rsvd	DLR2	Reserved	
R-0		R/W-0	R/W-1	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

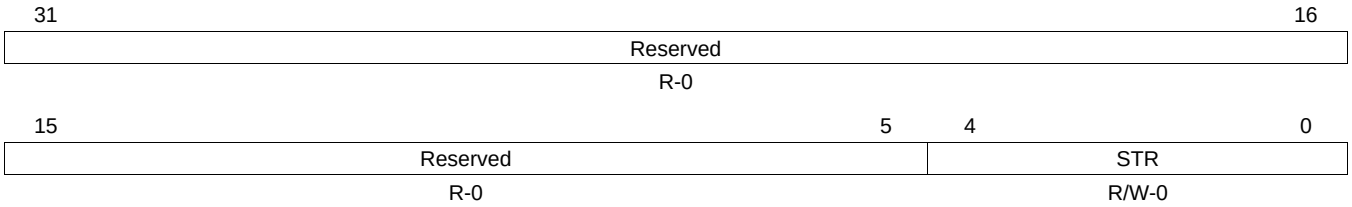
Table 6-3. Datalogger Register (DLR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Do not change these bits from their default value of 0.
4	DLR4	0-1	Configuration access: setting this bit allows the host processor to configure the PBIST controller registers.
3	Reserved	1	Reserved. Do not change this bit from its default value of 1.
2	DLR2	0-1	ROM-based testing: setting this bit enables the PBIST controller to execute test algorithms that are stored in the PBIST ROM.
1-0	Reserved	0	Reserved. Do not change these bits from their default value of 0.

6.5.3 Program Control Register (PCR)

Figure 6-5 and Table 6-4 illustrate this register.

Figure 6-5. Program Control Register (PCR) (offset = 16Ch)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-4. Program Control Register (PCR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	STR		PBIST Controller Mode
		0	Reserved
		1h	Start / Time Stamp mode restart
		2h	Resume / Emulation read
		4h	Stop
		8h	Step / Step for emulation mode
		10h	Check MISR mode
		All other values	Reserved

6.5.4 PBIST Activate/Clock Enable Register (PACT)

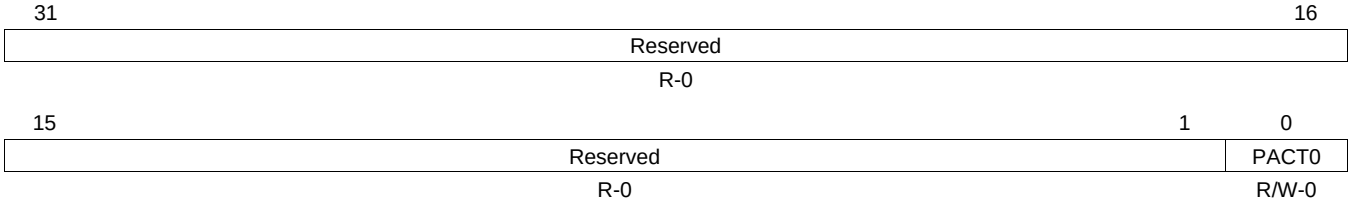
This is the first register that needs to be programmed to activate the PBIST controller. Bit [0] is used for static clock gating, and unless a 1 is written to this bit, all the internal PBIST clocks are shut off. Figure 6-6 and Table 6-5 illustrate this register.

- **PACT0**

This bit must be set to turn on internal PBIST clocks. Setting this bit asserts an internal signal that is used as the clock gate enable. As long as this bit is 0, any access to PBIST will not go through, and PBIST will remain in an almost zero-power mode.

NOTE: This register must be programmed to 1h during application self-test.

Figure 6-6. PBIST Activate/Clock Enable Register (PACT) (offset = 180h)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

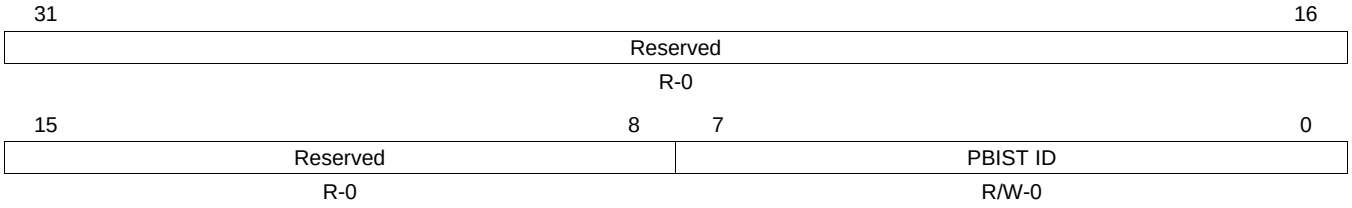
Table 6-5. PBIST Activate/Clock Enable Register (PACT) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	PACT0		PBIST internal clocks enable.
		0	Disable PBIST internal clocks.
		1	Enable PBIST internal clocks.

6.5.5 PBIST ID Register (PBISTID)

Functionality of the register is described in Figure 6-7 and Table 6-6.

Figure 6-7. PBIST ID Register (PBISTID) (offset = 184h)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-6. PBIST ID Register (PBISTID) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	PBIST ID	0-FFh	This is a unique ID assigned to each PBIST controller in a device with multiple PBIST controllers.

6.5.6 Override Register (OVER)

Functionality of the register is described in Figure 6-8 and Table 6-7.

- **OVER0**

While doing ROM-based testing, each algorithm downloaded from the ROM has a memory mask associated with it that defines the applicable memory groups the algorithm will be run on. By default, this bit is set to 1, which means the memory mask that is downloaded from the ROM will overwrite the RAM info registers. The override bit can be reset by writing a 0 to it. In this case, the application can select the RAM groups to be tested by configuring the RAM info registers.

NOTE: When this override bit = 0, each algorithm selected in the ALGO register runs on each RAM selected in the RINFOL and RINFOU registers. It must be ensured that:

1. Only the same type of memories (single port or two port) are selected, and
2. Only memories that are valid for all algorithms enabled via the ALGO register are selected.

If the above two requirements are not met, the memory self-test fails.

Figure 6-8. Override Register (OVER) (offset = 188h)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		Reserved		OVER0
R-0		R-0		R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-7. Override Register (OVER) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2-1	Reserved	0	Reserved. Do not change these bits from their default value of 0.
0	OVER0		RINFO Override Bit
		0	The RAM info registers RINFOL and RINFOU are used to select the memories for test.
		1	The memory information available from ROM overrides the RAM selection from the RAM info registers RINFOL and RINFOU.

6.5.7 Fail Status Fail Register (FSRF0)

This register indicates if Port0 failures occurred during a memory self-test. Bit [0] gets set whenever a failure occurs. Functionality of the register is described in Figure 6-9 and Table 6-8.

Figure 6-9. Fail Status Fail Register 0 (FSRF0) (offset = 190h)

31	16
Reserved	
R-0	

15	1	0
Reserved		FSRF0
R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 6-8. Fail Status Fail Register 0 (FSRF0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	FSRF0		Fail Status 0. This bit would be cleared by reset of the module using MSTGCR register in system module.
		0	No failure occurred.
		1	Failure occurred on port 0.

6.5.8 Fail Status Count Registers (FSRC0 and FSRC1)

These registers keep count of the number of failures observed during the memory self-test. The PBIST controller stops executing the memory self-test whenever a failure occurs in any memory instance for any of the test algorithms. The value in FSRC0 / FSRC1 gets incremented by one whenever a failure occurs and gets decremented by one when the failure is processed. FSRC0 is for Port 0 and FSRC1 is for Port 1. Figure 6-10 and Table 6-9 illustrate the FSRC0 register; Figure 6-11 and Table 6-10 illustrate the FSRC1 register.

Figure 6-10. Fail Status Count 0 Register (FSRC0) (offset = 198h)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		FSRC0	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 6-9. Fail Status Count 0 Register (FSRC0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	FSRC0	0-FFh	Fail Status Count 0. Indicates the number of failures on port 0.

Figure 6-11. Fail Status Count Register 1 (FSRC1) (offset = 19Ch)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		FSRC1	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 6-10. Fail Status Count Register 1 (FSRC1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	FSRC1	0-FFh	Fail Status Count 1. Indicates the number of failures on port 1.

6.5.9 Fail Status Address Registers (FSRA0 and FSRA1)

These registers capture the memory address of the first failure on port 0 and port 1, respectively. Figure 6-12 and Table 6-11 illustrate the FSRA0 register; Figure 6-13 and Table 6-12 illustrate the FSRA1 register.

Figure 6-12. Fail Status Address 0 Register (FSRA0) (offset = 1A0h)

31	16
Reserved	
R-0	
15	0
FSRA0	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 6-11. Fail Status Address 0 Register (FSRA0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FSRA0	0-FFFFh	Fail Status Address 0. Contains the address of the first failure.

Figure 6-13. Fail Status Address 1 Register (FSRA1) (offset = 1A4h)

31	16
Reserved	
R-0	
15	0
FSRA1	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 6-12. Fail Status Address 1 Register (FSRA1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FSRA1	0-FFFFh	Fail Status Address 1. Contains the address of the first failure.

6.5.10 Fail Status Data Registers (FSRDL0 and FSRDL1)

These registers are used to capture the failure data in case of a memory self-test failure. FSRDL0 corresponds to Port 0 and FSRDL1 corresponds to Port 1. Figure 6-14 and Table 6-13 illustrate the FSRDL0 register; Figure 6-15 and Table 6-14 illustrate the FSRDL1 register.

Figure 6-14. Fail Status Data Register 0 (FSRDL0) (offset = 1A8h)

31	16
FSRDL0	
R-AAAAh	

15	0
FSRDL0	
R-AAAAh	

LEGEND: R = Read only; -*n* = value after reset

Table 6-13. Fail Status Data Register 0 (FSRDL0) Field Descriptions

Bit	Field	Description
31-0	FSRDL0	Failure data on port 0

Figure 6-15. Fail Status Data Register 1 (FSRDL1) (offset = 1B0h)

31	16
FSRDL1	
R-AAAAh	

15	0
FSRDL1	
R-AAAAh	

LEGEND: R = Read only; -*n* = value after reset

Table 6-14. Fail Status Data Register 1 (FSRDL1) Field Descriptions

Bit	Field	Description
31-0	FSRDL1	Failure data on port 1

6.5.11 ROM Mask Register (ROM)

This two-bit register sets appropriate ROM access modes for the PBIST controller. This register is illustrated in Figure 6-16. It can be programmed according to Table 6-15.

Figure 6-16. ROM Mask Register (ROM) (offset = 1C0h)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ROM	
R-0		R/W-3h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-15. ROM Mask Register (ROM) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1-0	ROM		ROM Mask
		0	No information is used from ROM.
		1h	Only RAM Group information from ROM.
		2h	Only Algorithm information from ROM.
		3h	Both Algorithm and RAM Group information from ROM. This option should be selected for application self-test.

6.5.12 ROM Algorithm Mask Register (ALGO)

This register is used to enable the algorithms to be used for the memory self-test routine. Each bit corresponds to a specific algorithm. For an algorithm to be enabled, the corresponding bit in this register must be set to 1. The default value of this register is all 1s, which means all the algorithms are enabled. Bit [0] controls whether algorithm 1 is enabled or not. Figure 6-17 and Table 6-16 illustrate this register.

NOTE: Refer to Table 2-6 for available algorithms and the memories on which each algorithm can be run.

Figure 6-17. ROM Algorithm Mask Register (ALGO) (offset = 1C4h)

31	24	23	16
ALGO[31-24]		ALGO[23-16]	
R/W-FFh		R/W-FFh	

15	8	7	0
ALGO[15-8]		ALGO[7-0]	
R/W-FFh		R/W-FFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-16. Algorithm Mask Register (ALGO) Field Descriptions

Bit	Field	Value	Description
31-0	ALGO[*n*]		Algorithm 1-32 enable
		0	Algorithm *n* is not selected.
		1	Selects algorithm *n* for PBIST run.

6.5.13 RAM Info Mask Lower Register (RINFOL)

This register is used to select RAM groups to run the algorithms selected in the ALGO register. For an algorithm to be executed on a particular RAM group, the corresponding bit in this register must be set to 1. The default value of this register is all 1s, which means all the RAM groups are selected. Bit [0] controls whether RAM group 1 is selected or not. Figure 6-18 and Table 6-17 illustrate this register.

The information from this register is used only when bit [0] in the OVER register is not set.

NOTE: Refer to Table 2-5 for RAM info groups.

Figure 6-18. RAM Info Mask Lower Register (RINFOL) (offset = 1C8h)

31	24	23	16
RINFOL[31-24]		RINFOL[23-16]	
R/W-FFh		R/W-FFh	
15	**8**	**7**	**0**
RINFOL[15-8]		RINFOL[7-0]	
R/W-FFh		R/W-FFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-17. RAM Info Mask Lower Register (RINFOL) Field Descriptions

Bit	Field	Value	Description
31-0	RINFOL[*n*]		RAM Group 1-32 enable
		0	RAM Group *n* is not selected.
		1	Selects RAM group *n* for PBIST run.

6.5.14 RAM Info Mask Upper Register (RINFOU)

This register is used to select RAM groups to run the algorithms selected in the ALGO register. For an algorithm to be executed on a particular RAM group, the corresponding bit in this register must be set to 1. The default value of this register is all 1s, which means all the RAM groups are selected. Bit [0] controls whether RAM group 33 is selected or not. Figure 6-19 and Table 6-18 illustrate this register.

Figure 6-19. RAM Info Mask Upper Register (RINFOU) (offset = 1CCh)

31	24	23	16
RINFOU[31-24]		RINFOU[23-16]	
R/W-FFh		R/W-FFh	
15	**8**	**7**	**0**
RINFOU[15-8]		RINFOU[7-0]	
R/W-FFh		R/W-FFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6-18. RAM Info Mask Upper Register (RINFOU) Field Descriptions

Bit	Field	Value	Description
31-0	RINFOU[*n*]		RAM Group 33-64 enable
		0	RAM Group *n* is not selected.
		1	Selects RAM group *n* for PBIST run.

6.6 PBIST Configuration Examples

The following examples assume that the PLL is locked and selected as clock source with HCLK = VCLK = 80MHz.

6.6.1 Example 1 : Configuration of PBIST Controller to Run Self-Test on DCAN1 RAM

This example explains the configurations for running March13, Down1A and Map Column algorithms on DCAN1.

1. Program the HCLK to PBIST ROM clock ratio to 1:1 in system module.
   MSTGCR[9:8] = 0
2. Enable PBIST Controller in system module.
   MSIENA[31:0] = 0x00000001
3. Enable the PBIST self-test in system module.
   MSTGCR[3:0] = 0xA
4. Wait for at least 16 VCLK cycles in a software loop.
5. Enable the PBIST internal clocks.
   PACT = 0x1
6. Disable RAM Override. This will make the PBIST controller use the information provided by the application in the RINFOx and ALGO registers for the memory self-test.
   OVER = 0x0
7. Select the Algorithm (refer to Table 2-6).
   ALGO = 0x00000054 (Algo 3 = March13N, Algo 5 = down1A_red, Algo 7 = Map Column for two-port DCAN1 RAM)
8. Program the RAM group Info to select DCAN1 (DCAN1 RAM is Group 3, refer to Table 2-5).
   RINFOL = 0x00000004 (select RAM Group 3)
   RINFOU = 0x00000000 (since this device supports up to RAM Group 14)
9. Select both Algorithm and RAM information from on chip PBIST ROM.
   ROM = 0x3
10. Configure PBIST to run in ROM Mode and start PBIST run.
    DLR = 0x14
11. Wait for PBIST test to complete by polling MSTDONE bit in system module.
    while (MSTDONE != 1);
12. Once self-test is completed, check the Fail Status register FSRF0:
    a. In case there is a failure (FSRF0 = 0x01):
       i. Read RAMT register that indicates the RGS and RDS values of the failure RAM.
       ii. Read FSRC0 and FSRC1 registers that contains the failure count.
       iii. Read FSRA0 and FSRA1 registers that contains the address of first failure.
       iv. Read FSRDL0 and FSRDL1 registers that contains the failure data.
       v. Resume the Test, if required, using Program Control register (offset = 0x16C) STR = 2.
    b. In case there is no failure (FSRF0 = 0x00), the memory self-test is completed:
       i. Disable the PBIST internal clocks.
          PACT = 0
       ii. Disable the PBIST self-test.
          MSTGCR[3:0] = 0x5

6.6.2 Example 2 : Configuration of PBIST Controller to Run Self-Test on ALL RAM Groups

This example explains the configurations for running March13, Down1A and Map Column algorithms on all RAM groups defined in the PBIST ROM.

1. Program the HCLK to PBIST ROM clock ratio to 1:1 in system module.
   MSTGCR[9:8] = 0
2. Enable PBIST Controller in system module.
   MSIENA[31:0] = 0x00000001
3. Enable the PBIST self-test in system module.
   MSTGCR[3:0] = 0xA
4. Wait for at least 16 VCLK cycles in a software loop.
5. Enable the PBIST internal clocks.
   PACT = 0x1
6. Enable RAM Override.
   OVER = 0x1
7. Select the Algorithms to be run (refer to Table 2-6).
   ALGO = 0x000000FC (select March13N, Down1A and Map Column algorithms for single-port and two-port RAMs)
8. Select both Algorithm and RAM information from on chip PBIST ROM.
   ROM = 0x3
9. Configure PBIST to run in ROM Mode and kickoff PBIST test.
   DLR = 0x14
10. Wait for PBIST test to complete by polling MSTDONE bit in system module.
    while (MSTDONE != 1)
11. Once self-test is completed, check the Fail Status register FSRF0:
    a. In case there is a failure (FSRF0 = 0x01):
       i. Read RAMT register that indicates the RGS and RDS values of the failure RAM.
       ii. Read FSRC0 and FSRC1 registers that contains the failure count.
       iii. Read FSRA0 and FSRA1 registers that contains the address of first failure.
       iv. Read FSRDL0 and FSRDL1 registers that contains the failure data.
       v. Resume the Test, if required, using Program Control register (offset = 0x16C) STR = 2.
    b. In case there is no failure (FSRF0 = 0x00), the memory self-test is completed:
       i. Disable the PBIST internal clocks.
          PACT = 0
       ii. Disable the PBIST self-test.
          MSTGCR[3:0] = 0x5

CPU Self-Test Controller (STC) Module

This chapter describes the basics and configuration of the CPU self-test controller (STC) in the device.

Topic **Page**

7.1 General Description

The CPU self-test controller (STC) is used to test the ARM-CPU core using the Deterministic Logic Built-in Self-Test (LBIST) Controller as the test engine. To achieve better coverage for the self-test of complex cores like Cortex-R4, on-chip logic BIST is the preferred solution.

7.1.1 CPU Self-Test Controller Features

The CPU self-test controller has the following features:

- Capable of running the complete test as well as running a few intervals at a time
  - Ability to continue from the last executed interval (test set) as well as the ability to restart from the beginning (first test set)
  - Total of 26 intervals supported in this device
- Complete isolation of the self-tested CPU core from the rest of the system during the self-test run
  - The self-tested CPU core master bus transaction signals are configured to be in idle mode during the self-test run
  - Any master access to the CPU core under self-test will be held until the completion of the self-test
- Ability to capture the failure interval number
- Timeout counter for the CPU self-test run as a fail-safe feature
- Able to read the MISR data (shifted from LBIST controller) of the last executed interval of the self-test run for debugging purposes
- STCCLK determines the self-test execution speed, STC clock divider (STCCLKDIV) register in the system module is used to divide HCLK (system clock) to generate STCCLK

7.1.2 STC Block Diagram

STC module provides an interface to the LBIST controller implemented on the core.

The CPU STC is composed of following blocks of logic:

- ROM Interface
- FSM and Sequence Control
- Register Block
- Peripheral Bus Interface (VBUSP Interface)
- STC Bypass/ATE Interface

7.1.2.1 ROM Interface

This block handles the ROM address and control signal generation to read the self-test microcode from the ROM. The test microcode and golden signature value for each interval are stored in ROM.

7.1.2.1.1 FSM and Sequence Control

This block generates the signals and data to the LBIST controller based on the seed, test_type and scan chain depth.

7.1.2.1.2 Clock Control

The CLOCK CNTRL sub-block handles the internal clock selection and clock generation for the ROM and LBIST controller.

7.1.2.2 Register Block

This block handles the control of the self-test controller. This block contains various configuration and status registers which provide the result of a self-test run. These registers are memory mapped and accessible through the Peripheral Bus (VBUSP) Interface. This block controls the reseeding (reloading the existing seed of the PRPG) in the LBIST controller.

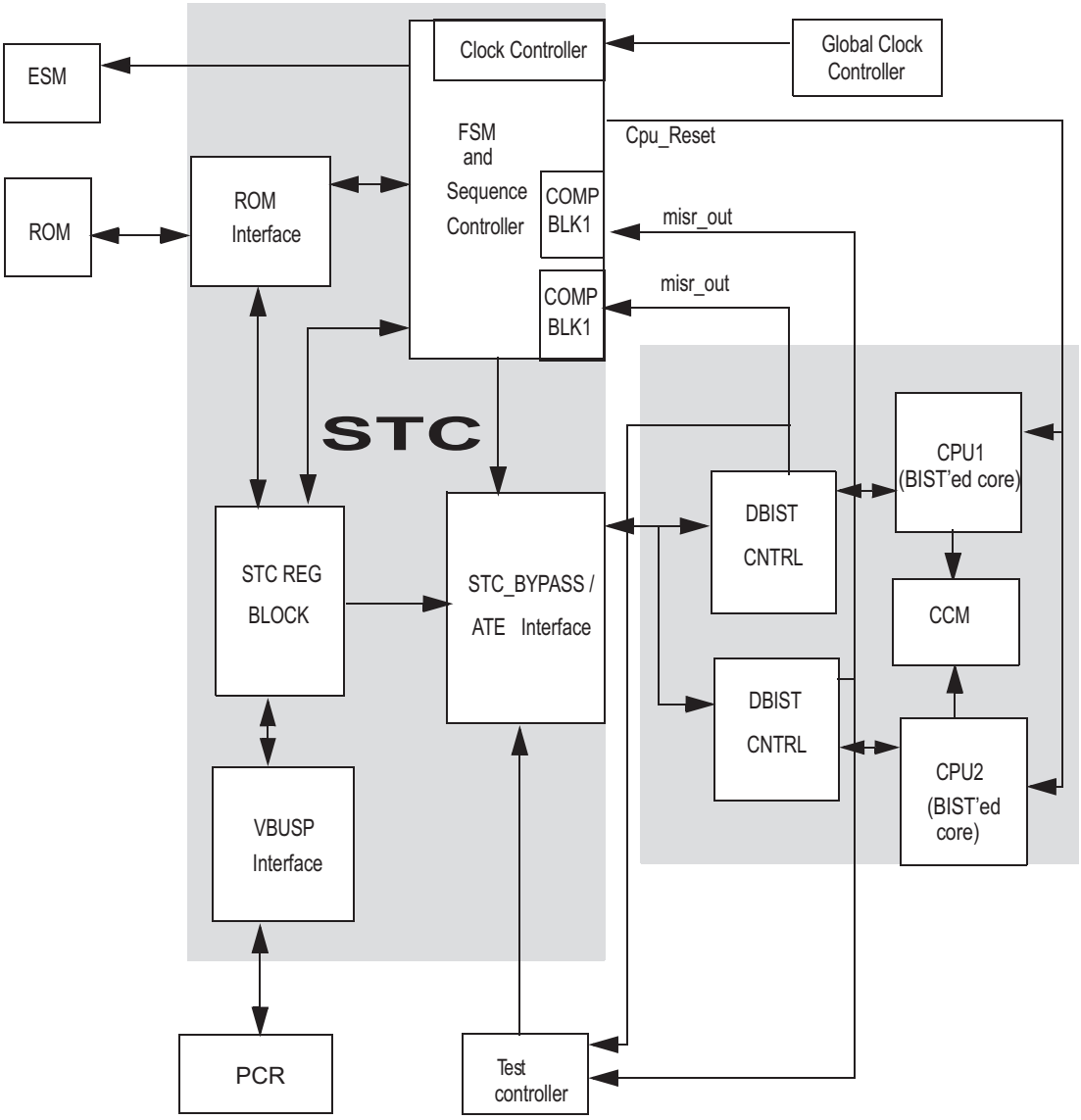
7.1.2.3 STC Bypass / ATE Interface

This is a production test interface. Only for TI internal use.

7.1.2.4 Peripheral Bus (VBUSP) Interface

STC control registers are accessed through the Peripheral Bus (VBUSP) Interface. During application programming, configuration registers are programmed through the Peripheral Bus Interface to enable and run the self-test controller.

Figure 7-1. STC Block Diagram



7.2 Application Self-Test Flow

This section describes the STC module configuration and the application self-test flow that the user should follow for successful execution.

The following two configurations must be part of the STC initialization code:

- STC clock rate configuration, STC clock divider (STCCLKDIV) register in system module is used to divide HCLK (system clock) to generate STCCLK
- Clear SYSESR register before triggering an STC test

7.2.1 STC Module Configuration

- Configure the test interval count using STCGCR0[31:16] register. A maximum of 26 intervals are supported in the device. The user can run 26 intervals together or in slices. If the tests are ran in slices, the user software can specify to the self-test controller whether to continue the run from the next interval onwards or to restart from interval 0 using bit STCGCR0[0]. This bit gets reset after the completion of the self-test run.
- Configure self-test run timeout counter preload register STCTPR. This register contains the total number of VBUS clock cycles it will take before a self-test timeout error (TO_ERR) will be triggered after the initiation of the self-test run.
- Enable CPU self-test by writing the enable key to STCGCR1 register.

7.2.2 Context Saving

STC generates a CPU reset after completion of the test regardless of pass or fail. You can run the STC test during startup or can divide STC into 26 or fewer intervals and run them during normal operation.

If STC is ran only on startup, the user software need not save the CPU contents since the reset caused will go through all startup configurations. User should check the STCGSTAT register for the self-test status before going to the application software.

If STC is divided into intervals and ran, user software must save the CPU contents and reload them after the CPU reset caused by the completion of the STC test interval. The check for STC status should bypass STC run if the reset is caused by an STC run to prevent a cyclic reset, that is, if reset is caused by STC the second time through, then it should not be ran again. The user should also check the STCGSTAT register for the self-test status before restoring the application software.

Following are some of the registers that are required to be backed up before and restored after self-test:

1. CPU core registers (all modes R0-R15, PC, CPSR)
2. CP15 System Control Coprocessor registers - MPU control and configuration registers, Auxiliary Control Register used to Enable ECC, Fault Status Register.
3. CP13 Coprocessor Registers - FPU configuration registers, General Purpose Registers
4. Hardware Break Point and watch point registers like BVR, BSR, WVR, WSR etc.

For more information on the CPU reset, refer to the ARM® Cortex®-R4 Technical Reference Manual.

NOTE: Check all reset source flags in the SYSESR register after a CPU BIST execution. If a flag in addition to CPU reset is set, clear the CPU reset flag and service the other reset sources accordingly.

7.2.3 Entering CPU Idle Mode

After enabling the STC test by writing the STC enable key, the test is triggered only after the CPU is taken to idle mode by executing the CPU Idle Instruction **asm(“ WFI”).**

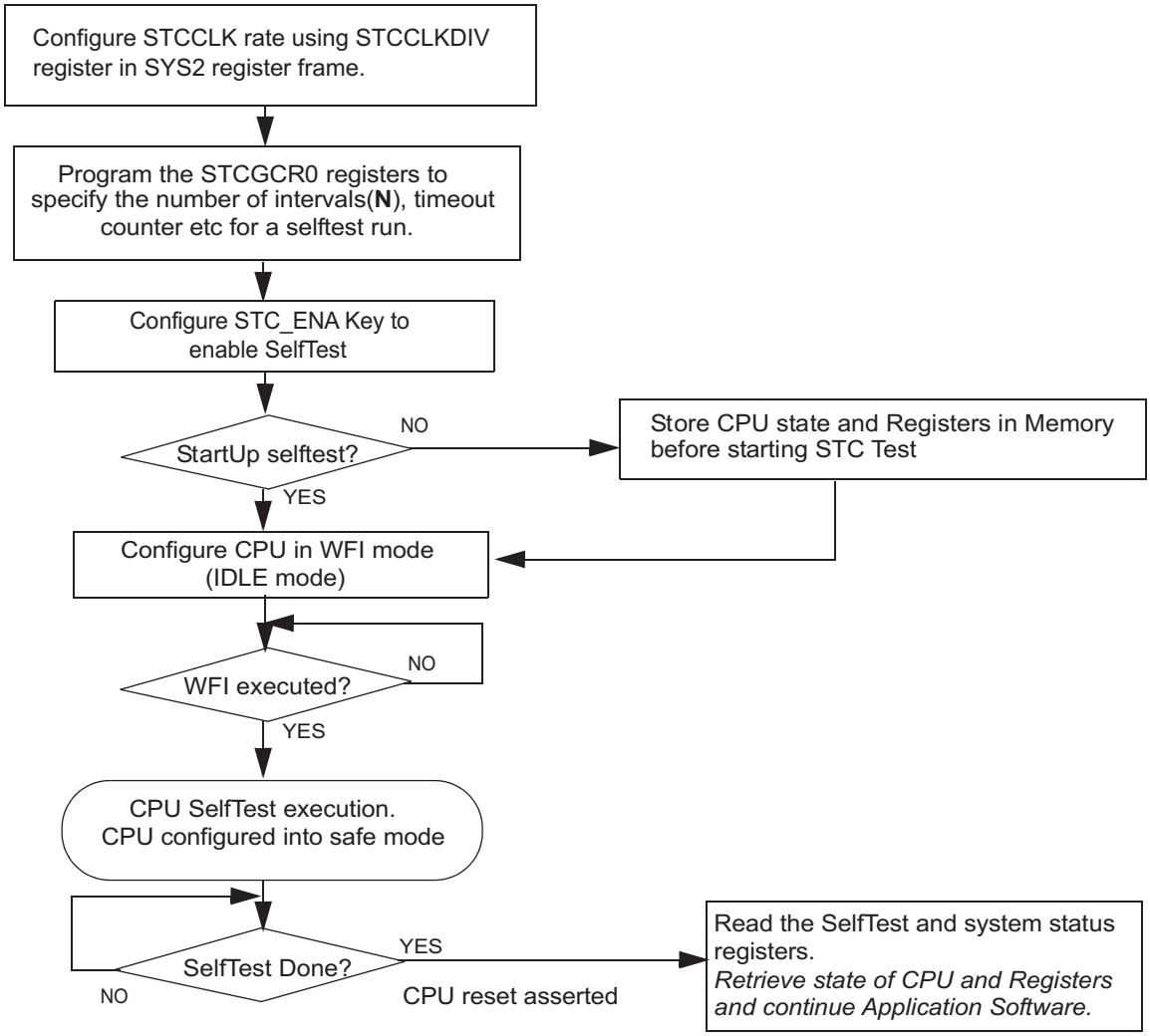
7.2.4 Self-Test Completion and Error Generation

At the end of each interval, the 128 bit MISR value (reflected in registers CPUx_CURMISR[3:0]) from the DBIST controller is shifted into the STC. This is compared with the golden MISR value stored in the ROM.

At the end of a CPU self-test, the STC controller updates the status flags in the Global Status Register (STCGSTAT) and resets the CPU. In case of a MISR mismatch or a test timeout, an error is generated through the ESM module. A test error signal is asserted when a MISR mis-compare occurs during the self-test. A time out error is asserted when a timeout occurs during the self-test, meaning the test could not complete within the time specified in the timeout counter preload register STCTPR. However, at the device level, these two errors are combined and mapped to a single ESM channel. To identify which error occurred, user software must check the Global Status Register (STCGSTAT) and the Fail Status Register (STCFSTAT) in the ESM interrupt service routine.

Figure 7-2 illustrates the application self-test flow chart, it is drawn based on the assumption that the device has gone through startup, necessary clocks initialized and the SYSESR register bits cleared.

Figure 7-2. Application Self-Test Flow Chart



7.3 STC Test Coverage and Duration

The test coverage and number of test execution cycles (STCCLK) for each test interval when the device is running at HCLK = 90 MHz, VCLK = 90 MHz, and STCCLK = 45 MHz are shown in Table 7-1.

Table 7-1. STC Test Coverage and Duration

Intervals	Test Coverage (%)	Test Time (Cycles)	Test Time (µs)
0	0	0	0.00
1	60.06	1365	30.33
2	68.71	2730	60.67
3	73.35	4095	91.00
4	76.57	5460	121.33
5	78.7	6825	151.67
6	80.4	8190	182.00
7	81.76	9555	212.33
8	82.94	10920	242.67
9	83.84	12285	273.00
10	84.58	13650	303.33
11	85.31	15015	333.67
12	85.9	16380	364.00
13	86.59	17745	394.33
14	87.17	19110	424.67
15	87.67	20475	455.00
16	88.11	21840	485.33
17	88.53	23205	515.67
18	88.93	24570	546.00
19	89.26	25935	576.33
20	89.56	27300	606.67
21	89.86	28665	637.00
22	90.1	30030	667.33
23	90.36	31395	697.67
24	90.62	32760	728.00
25	90.86	34125	758.33
26	91.06	35490	788.67

7.4 STC Control Registers

STC control registers are accessed through Peripheral Bus (VBUSP) interface. Read and write access in 8, 16, and 32 bits are supported. The base address for the control registers is FFFF E600h.

NOTE: In suspend mode, all registers can be written regardless of user or privilege mode.

Table 7-2. STC Control Registers

Offset	Acronym	Register Description	Section
00h	STCGCR0	STC Global Control Register 0	Section 7.4.1
04h	STCGCR1	STC Global Control Register 1	Section 7.4.2
08h	STCTPR	Self-Test Run Timeout Counter Preload Register	Section 7.4.3
0Ch	STC_CADDR	STC Current ROM Address Register	Section 7.4.4
10h	STCCICR	STC Current Interval Count Register	Section 7.4.5
14h	STCGSTAT	Self-Test Global Status Register	Section 7.4.6
18h	STCFSTAT	Self-Test Fail Status Register	Section 7.4.7
1Ch	CPU1_CURMISR3	CPU1 Current MISR Register 3	Section 7.4.8
20h	CPU1_CURMISR2	CPU1 Current MISR Register 2	Section 7.4.8
24h	CPU1_CURMISR1	CPU1 Current MISR Register 1	Section 7.4.8
28h	CPU1_CURMISR0	CPU1 Current MISR Register 0	Section 7.4.8
2Ch	CPU2_CURMISR3	CPU2 Current MISR Register 3	Section 7.4.9
30h	CPU2_CURMISR2	CPU2 Current MISR Register 2	Section 7.4.9
34h	CPU2_CURMISR1	CPU2 Current MISR Register 1	Section 7.4.9
38h	CPU2_CURMISR0	CPU2 Current MISR Register 0	Section 7.4.9
3Ch	STCSCSCR	Signature Compare Self Check Register	Section 7.4.10

7.4.1 STC Global Control Register 0 (STCGCR0)

This register is described in Figure 7-3 and Table 7-3.

NOTE: On a power-on reset or system reset, this register gets reset to its default values.

Figure 7-3. STC Global Control Register 0 (STCGCR0) (offset = 00h)

31	16
INTCOUNT	
R/W-1	

15	1	0
Reserved		RS_CNT
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; *-n* = value after reset

Table 7-3. STC Global Control Register 0 (STCGCR0) Field Descriptions

Bit	Field	Value	Description
31-16	INTCOUNT	0-FFFFh	Number of intervals of self-test run. This register specifies the number of intervals to run for the self-test run. This corresponds to the number of intervals to be ran from the value reflected in the current interval counter.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	RS_CNT		Restart or Continue This bit specifies whether to continue the run from next interval onwards or to restart from interval 0. This bit gets reset after the completion of a self-test run.
		0	Continue STC run from the previous interval.
		1	Restart STC run from interval 0.

7.4.2 STC Global Control Register 1 (STCGCR1)

This register is described in Figure 7-4 and Table 7-4.

NOTE: On a power-on reset or system reset, this register resets to its default value. Also at the completion of a self-test run, this register automatically resets to its default value.

Figure 7-4. STC Global Control Register 1 (STCGCR1) (offset = 04h)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		STC_ENA	
R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; *-n* = value after nPORST (power on reset) or System reset

Table 7-4. STC Global Control Register 1 (STCGCR1) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	STC_ENA		Self-test run enable key.
		Ah	Self-test run is enabled.
		All other values	Self-test run is disabled.

7.4.3 Self-Test Run Timeout Counter Preload Register (STCTPR)

This register is described in Figure 7-5 and Table 7-5.

NOTE: On a power-on reset or system reset, this register gets reset to its default values.

Figure 7-5. Self-Test Run Timeout Counter Preload Register (STCTPR) (offset = 08h)

31	0
RTOD	
R/WP-FFFF FFFFh	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after nPORST (power on reset) or System reset

Table 7-5. Self-Test Run Timeout Counter Preload Register (STCTPR)

Bit	Field	Description
31-0	RTOD	Self-test timeout count preload. This register contains the total number of VBUS clock cycles it will take before a self-test timeout error (TO_ERR) will be triggered after the initiation of the self-test run. This is a fail safe feature to prevent the device from hanging up due to a runaway test during the self-test execution. The preload count value gets loaded into the self-test time out down counter whenever a self-test run is initiated (STC_KEY is enabled) and gets disabled on completion of a self-test run.

7.4.4 STC Current ROM Address Register (STC_CADDR)

This register is described in Figure 7-6 and Table 7-6.

NOTE: When the RS_CNT bit in STCGCR0 is set to a 1 on the start of a self-test run, or on a power-on reset or system reset, this register resets to all zeroes.

Figure 7-6. STC Current ROM Address Register (STC_CADDR) (offset = 0Ch)

31	0
ADDR	
R-0	

LEGEND: R = Read only; -*n* = value after nPORST (power on reset) or System reset

Table 7-6. STC Current ROM Address Register (STC_CADDR) Field Descriptions

Bit	Field	Description
31-0	ADDR	Current ROM Address This register reflects the current ROM address (for micro code load) which is the current value of the STC program counter.

7.4.5 STC Current Interval Count Register (STCCICR)

This register is described in Figure 7-7 and Table 7-7.

NOTE: When the RS_CNT bit in STCGCR0 is set to a 1 or on a power-on reset, the current interval counter resets to the default value.

Figure 7-7. STC Current Interval Count Register (STCCICR) (offset = 10h)

31	16
Reserved	
R-0	

15	0
N	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-7. STC Current Interval Count Register (STCCICR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	N	0-FFFFh	Interval Number This specifies the last executed interval number.

7.4.6 Self-Test Global Status Register (STCGSTAT)

This register is described in Figure 7-8 and Table 7-8.

NOTE: The two status bits can be cleared to their default values on a write of 1 to the bits. Additionally, when the STC_ENA key in STCGCR1 is written from a disabled state to enabled state, the two status flags get cleared to their default values. This register gets reset to its default value with power-on reset assertion.

Figure 7-8. Self-Test Global Status Register (STCGSTAT) (offset = 14h)

31	8
Reserved	
R-0	

2	1	0
Reserved	TEST_FAIL	TEST_DONE
R-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; *-n* = value after reset

Table 7-8. Self-Test Global Status Register (STCGSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	TEST_FAIL		Test Fail
		0	Self-test run has not failed.
		1	Self-test run has failed.
0	TEST_DONE		Test Done
		0	Not completed.
		1	Self-test run completed.
			The test done flag is set to a 1 for any of the following conditions: 1. When the STC run is complete without any failure 2. When a failure occurs on a STC run 3. When a timeout failure occurs Reset is generated to the CPU on which the STC run is being performed when TEST_DONE goes high (the test is completed).

7.4.7 Self-Test Fail Status Register (STCFSTAT)

This register is described in Figure 7-9 and Table 7-9.

NOTE: The three status bits can be cleared to their default values on a write of 1 to the bits. Additionally, when the STC_ENA key in STCGCR1 is written from a disabled state to an enabled state, the three status bits get cleared to their default values. This register gets reset to its default value with power-on reset assertion.

Figure 7-9. Self-Test Fail Status Register (STCFSTAT) (offset = 18h)

31	8
Reserved	
R-0	

3	2	1	0
Reserved	TO_ERR	CPU2_FAIL	CPU1_FAIL
R-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; *-n* = value after nPORST (power on reset) or System reset

Table 7-9. Self-Test Fail Status Register (STCFSTAT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TO_ERR		Timeout Error
		0	No time out error occurred.
		1	Self-test run failed due to a timeout error.
1	CPU2_FAIL		CPU2 failure info
		0	No MISR mismatch for CPU2.
		1	Self-test run failed due to MISR mismatch for CPU2.
0	CPU1_FAIL		CPU1 failure info
		0	No MISR mismatch for CPU1.
		1	Self-test run failed due to MISR mismatch for CPU1.

7.4.8 CPU1 Current MISR Registers (CPU1_CURMISR3-CPU1_CURMISR0)

This register is described in Figure 7-10 through Figure 7-13 and Table 7-10.

NOTE: On a power-on reset or system reset, this register gets reset to its default values.

Figure 7-10. CPU1 Current MISR Register 3 (CPU1_CURMISR3) (offset = 1Ch)

31	16
MISR[31:16]	
R-0	

15	0
MISR[15:0]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 7-11. CPU1 Current MISR Register 2 (CPU1_CURMISR2) (offset = 20h)

31	16
MISR[63:48]	
R-0	

15	0
MISR[47:32]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 7-12. CPU1 Current MISR Register 1 (CPU1_CURMISR1) (offset = 24h)

31	16
MISR[95:80]	
R-0	

15	0
MISR[79:64]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 7-13. CPU1 Current MISR Register 0 (CPU1_CURMISR0) (offset = 28h)

31	16
MISR[127:112]	
R-0	

15	0
MISR[111:96]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-10. CPU1 Current MISR Register (CPU1_CURMISR[3:0]) Field Descriptions

Bit	Field	Description
127-0	MISR	MISR data from CPU1 This register contains the MISR data from the CPU1 for the most recent interval. This value is compared with the GOLDEN MISR value copied from ROM.

7.4.9 CPU2 Current MISR Registers (CPU2_CURMISR3-CPU2_CURMISR0)

This register is described in Figure 7-14 through Figure 7-17 and Table 7-11.

NOTE: On a power-on reset or system reset, this register gets reset to its default values.

Figure 7-14. CPU2 Current MISR Register 3 (CPU2_CURMISR3) (offset = 2Ch)

31	16
MISR[31:16]	
R-0	

15	0
MISR[15:0]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 7-15. CPU2 Current MISR Register 2 (CPU2_CURMISR2) (offset = 30h)

31	16
MISR[63:48]	
R-0	

15	0
MISR[47:32]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 7-16. CPU2 Current MISR Register 1 (CPU2_CURMISR1) (offset = 34h)

31	16
MISR[95:80]	
R-0	

15	0
MISR[79:64]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 7-17. CPU2 Current MISR Register 0 (CPU2_CURMISR0) (offset = 38h)

31	16
MISR[127:112]	
R-0	

15	0
MISR[111:96]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7-11. CPU2 Current MISR Register (CPU2_CURMISR[3:0]) Field Descriptions

Bit	Field	Description
127-0	MISR	MISR data from CPU2 This register contains the MISR data from the CPU2 for the most recent interval. This value is compared with the GOLDEN MISR value copied from ROM.

7.4.10 Signature Compare Self Check Register (STCSCSCR)

This register is described in Figure 7-18 and Table 7-12.

NOTE: This register is used to enable the self-check feature of the CPU Self-Test Controller's (STC) signature compare logic. Self-check can only be done for the STC interval 0 by setting the RS_CNT bit in STCGCR0 to 1 to restart the self-test. The STC run will fail for signature miss-compare, provided the signature compare logic is operating correctly. To proceed with regular CPU self-test, STCSCSCR should be programmed to disable the self-check feature and clear the RS_CNT bit in STCGCR0 to 0. This register gets reset to its default value with power-on or system reset assertion.

Figure 7-18. Signature Compare Self Check Register (STCSCSCR) (offset = 3Ch)

31	15
Reserved	
R-0	

15	5	4	3	0
Reserved		FAULT_INS	SELF_CHECK_KEY	
R-0		R/WP-0	R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after nPORST (power on reset) or System reset

Table 7-12. Signature Compare Self Check Register (STCSCSCR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4	FAULT_INS		Fault Insertion
		0	No fault insertion.
		1	Generates a signal out of the STC module to the CPU for inserting a stuck-at fault (stuck-at-0) in the CPU which will make the signature compare fail.
3-0	SELF_CHECK_KEY		Signature compare logic self-check key
		Ah	Signature compare self-check is enabled.
		All other values	Signature compare self-check is disabled.

7.5 STC Configuration Example

The following example assumes that the PLL is locked and selected as the system clock source with VCLK = HCLK = 80MHz.

7.5.1 Example 1: Self-Test Run for 24 Interval

This example explains the configurations for running STC Test for maximum Test Intervals 24.

1. Maximum STC clock rate support is HCLK/2. STCCLKDIV[26:24] register in the secondary system module frame at location 0xFFFF E108 is used to program the STCCLK prescaler to divide down from HCLK as necessary (0 for /1, 1 for /2, 2 for /3, and so on).
2. Clear CPU RST status bit in the System Exception Status Register in the system module.
   SYSESR[5] = 1
3. Configure the test interval count in the STC module.
   STCGCR0[31:16] = 24
4. Configure the self-test run rime out counter preload Register. This will be set to the maximum time for this example but can be calculated as t_{TO} = STCPTR X $t_{c(VCLK)}$
   STCTPR[31:0] = 0xFFFFFFFF
5. Enable CPU self-test.
   STCGCR1[3:0] = 0xA
6. Perform a context save of CPU state and configuration registers that gets reset on CPU reset.
7. Put the CPU in idle mode by executing CPU Idle Instruction.
   asm(“ WFI”)
8. Upon CPU reset, verify the CPU RST status bit in the System Exception Status Register is set. This also verifies that no other resets occurred during the self-test.
   SYSESR[5] == 1
9. Check the STCGSTAT register for the self-test Status.
   Check the TEST_DONE bit before evaluating the TEST_FAIL bit.

If TEST_DONE = 0, the self-test is not completed. Restart/resume the STC test.

If (TEST_DONE = 1 and TEST_FAIL = 1), the self-test is completed and failed.

- Read the STC Fail Status Register STCFSTAT[2:0] to identify the type of Failure (Timeout, CPU1 fail, CPU2 fail).

In case there is no failure (TEST_DONE = 1 and TEST_FAIL = 0), the memory self-test is completed successfully.

- Recover the CPU status, configuration registers and continue application software.

CPU Compare Module for Cortex™-R4 (CCM-R4)

This chapter describes the CPU compare module for Cortex-R4 (CCM-R4). This device implements two instances of the Cortex-R4 CPU which are running in lock step to detect faults which may result in unsafe operating conditions. The CCM-R4 detects faults and signals them to an error signaling module (ESM).

Topic **Page**

8.1 Main Features

Safety-critical applications require run-time detection of faults in the Central Processing Unit (CPU). For this purpose, the CPU Compare Module for Cortex-R4 (CCM-R4) compares the core compare bus outputs of two Cortex-R4 CPUs running in a 1oo1D (one-out-of-one, with diagnostics) lockstep configuration. Any difference in the core compare bus outputs of the CPUs is flagged as an error. For diagnostic purposes, the CCM-R4 also incorporates a self-test capability to allow for boot time checking of hardware faults within the CCM-R4 itself.

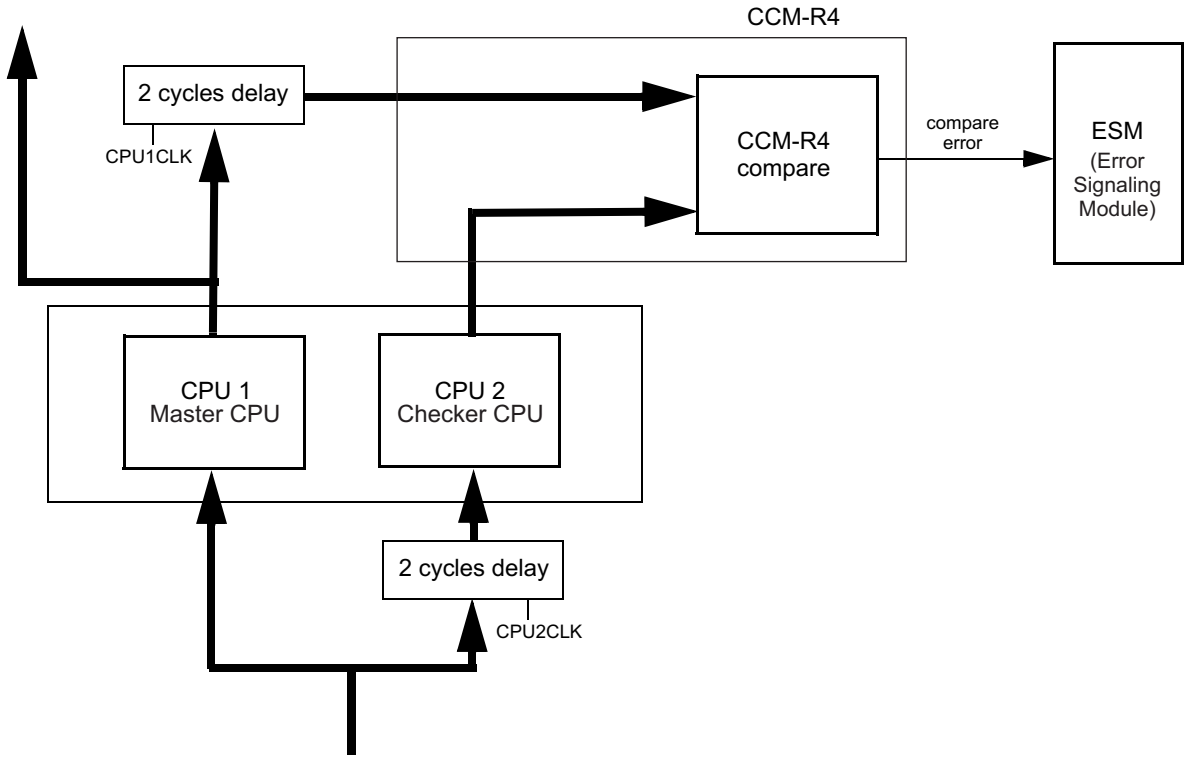
The main features of the CCM-R4 are:

- run-time detection of faults
- self-test capability
- error forcing capability

8.2 Block Diagram

Figure 8-1 shows the interconnection diagram of the CCM-R4 with the two Cortex-R4 CPUs. The core compare bus outputs of the CPUs are compared in the CCM-R4. To avoid common mode impacts, the signals of the CPUs to be compared are temporally diverse. The output signals of the master CPU are delayed 2 cycles while the input signals of checker CPU are delayed 2 cycles.

Figure 8-1. Block Diagram



8.3 Module Operation

The CCM-R4 compares the core compare bus outputs of the master and checker Cortex-R4 CPUs on the microcontroller and signals an error on any mismatch. This comparison is started 6 CPU clock cycles after the CPU comes out of reset to ensure that CPU output signals have propagated to a known value after reset. Once comparison is started, the CCM module continues to monitor the outputs of two CPUs without any software intervention. Upon an error software needs to handle it.

The CCM-R4 can run in one of the following four operating modes:

1. 1oo1D lock step
2. self-test
3. error forcing
4. self-test error forcing

The operating mode can be selected by writing a dedicated key to the key register (MKEY).

8.3.1 1oo1D Lock Step Mode

This is the default mode on start-up.

In lock step mode, the compare bus output signals of both CPUs are compared. A difference in the CPU compare bus outputs is indicated by signaling an error to the ESM which sets the error flag “CCM-R4 - compare”.

NOTE: The CPU compare error asserts “CCM-R4 self-test error” flag as well. By doing this, the CPU compare error has two paths (“CCM-R4 - compare” and “CCM-R4 self-test error” flag) to the ESM, so that even if one of the paths fails, the error is still propagated to the ESM.

Not all internal registers of the Cortex R4 CPU have fixed values upon reset. To avoid an erroneous CCMR4 compare error, the application software needs to ensure that the CPU registers of both CPUs are initialized with the same values before the registers are used, including function calls where the register values are pushed onto the stack.

8.3.2 Self-Test Mode

In self-test mode, the CCM-R4 checks itself for faults. During self-test, the compare error module output signal is deactivated. Any fault detected inside the CCM-R4 will be flagged by ESM error “CCM-R4 - self-test”.

In self-test mode, the CCM-R4 automatically generates test patterns to look for any hardware faults. If a fault is detected, then a self-test error flag is set, a self-test error signal is asserted and sent to the ESM, and the self-test is terminated immediately. If no fault is found during self-test, the self-test complete flag is set. In both cases, the CCM-R4 remains in self-test mode after the test has been terminated or completed, and the application needs to switch the CCM-R4 mode by writing another key to the mode key register (MKEY). During the self-test operation, the compare error signal output to the ESM is inactive irrespective of the compare result.

There are two types of patterns generated by CCM-R4 during self-test mode:

i. Compare Match Test

ii. Compare Mismatch Test

CCM-R4 first generates Compare Match Test patterns, followed by Compare Mismatch Test patterns. Each test pattern is applied on both CPU signal inputs of the CCM-R4’s compare block and clocked for one cycle. The duration of self-test is 3615 CPU clock cycles (GCLK).

NOTE: During self-test, both CPUs can execute normally, but the compare logic will not be checking any CPU signals. Also during self-test, only the compare unit logic is tested and not the memory mapped register controls for the CCM-R4. The self-test is not interruptible.

8.3.2.1 Compare Match Test

During the Compare Match Test, there are four different test patterns generated to stimulate the CCM-R4. An identical vector is applied to both input ports at the same time expecting a compare match. These patterns cause the self-test logic to exercise every CPU compare bus output signal in parallel. If the compare unit produces a compare mismatch then the self-test error flag is set, the self-test error signal is generated, and the Compare Match Test is terminated.

The four test patterns used for the Compare Match Test are:

- All 1s on both CPU signal ports
- All 0s on both CPU signal ports
- 0xAs on both CPU signal ports
- 0x5s on both CPU signal ports

These four test patterns will take four clock cycles to complete. Table 8-1 illustrates the sequence of Compare Match Test.

Table 8-1. Compare Match Test Sequence

CPU 1 Signal Position									CPU 2 Signal Position									Cycle
n:8	7	6	5	4	3	2	1	0	n:8	7	6	5	4	3	2	1	0	
1s	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	0
0s	0	0	0	0	0	0	0	0	0s	0	0	0	0	0	0	0	0	1
0xA	1	0	1	0	1	0	1	0	0xA	1	0	1	0	1	0	1	0	2
0x5	0	1	0	1	0	1	0	1	0x5	0	1	0	1	0	1	0	1	3

8.3.2.2 Compare Mismatch Test

During the Compare Mismatch Test, the number of test patterns is equal to twice the number of CPU output signals to compare in lock step mode. An all 1s vector is applied to the CCM-R4's CPU1 input port and the same pattern is also applied to the CCM-R4's CPU2 input port but with one bit flipped starting from signal position 0. The un-equal vector will cause the CCM-R4 to expect a compare mismatch at signal position 0, if the CCM-R4 logic is working correctly. If, however, the CCM-R4 logic reports a compare match, the self-test error flag is set, the self-test error signal is asserted, and the Compare Mismatch Test is terminated.

This Compare Mismatch Test algorithm repeats in a domino fashion with the next signal position flipped while forcing all other signals to logic level 1. This sequence is repeated until every single signal position is verified on both CPU signal ports.

The Compare Mismatch Test is terminated if the CCM-R4 reports a compare match versus the expected compare mismatch. This test ensures that the compare unit is able to detect a mismatch on every CPU signal being compared. Table 8-2 illustrates the sequence of Compare Mismatch Test. There is no error signal is sent to ESM if the expected errors are seen with each pattern.

Table 8-2. Compare Mismatch Test Sequence

CPU 1 Signal Position											CPU 2 Signal Position										Cycle
n	n-1:8		7	6	5	4	3	2	1	0	n	n-1:8		7	6	5	4	3	2	1	0	
1	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	0	0
1	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	0	1	1
1	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	0	1	1	2
1	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	0	1	1	1	3
::																						
1	1	1s	1	1	1	1	1	1	1	1	1	0	1s	1	1	1	1	1	1	1	1	n-1
1	1	1s	1	1	1	1	1	1	1	1	0	1	1s	1	1	1	1	1	1	1	1	n
1	1	1s	1	1	1	1	1	1	1	0	1	1	1s	1	1	1	1	1	1	1	1	n+1
1	1	1s	1	1	1	1	1	1	0	1	1	1	1s	1	1	1	1	1	1	1	1	n+2
1	1	1s	1	1	1	1	1	0	1	1	1	1	1s	1	1	1	1	1	1	1	1	n+3
1	1	1s	1	1	1	1	0	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	n+4
::																						
1	0	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	2n-1
0	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	2n

8.3.3 Error Forcing Mode

In error forcing mode, a test pattern is applied to the CPU related inputs of the CCM-R4 compare logic to force an error in the compare error output signal of the compare unit. The ESM error flag “CCM-R4 - compare” is expected after the error forcing mode completes. As a side effect, the “CCM-R4 self-test error” flag is also asserted whenever the CPU compare error is asserted.

Error forcing mode is similar to the Compare Mismatch Test operation of self-test mode in which an unequal vector is applied to the CCM-R4 CPU signal ports. The error forcing mode forces the compare mismatch to actually assert the compare error output signal. This ensures that faults in the path between CCM-R4 and ESM is detected.

Only one hardcoded test pattern is applied into CCM-R4 during error forcing mode. A repeated 0x5 pattern is applied to CPU1 signal port of CCM-R4 input while a repeated 0xA pattern is applied to the CPU2 signal port of CCM-R4 input. The error forcing mode takes one cycle to complete. Hence, the failing signature is presented for one clock cycle. After that, the mode is automatically switched to lock step mode. The key register (MKEY) will indicate the lock step key mode once it is switched to lock step mode. During the one cycle required by the error forcing test, the CPU output signals are not compared. User should expect the ESM to trigger a response (report the CCM-R4 fail). If no error is detected by ESM, then a hardware fault is present.

8.3.4 Self-Test Error Forcing Mode

In self-test error forcing mode, an error is forced at the self-test error signal. The compare unit is still running in lockstep mode and the key is switched to lockstep after one clock cycle. The ESM error flag “CCM-R4 - self-test” is expected after the self-test error forcing mode completes. Once the expected errors are seen, the application can clean the error through ESM module.

8.3.5 Operation During CPU Debug Mode

Certain debug operations place the CPU in a halting debug state where the code execution is halted. Because halting debug events are asynchronous, there is a possibility for the debug requests to cause loss of lockstep. CCM-R4 will disable upon detection of halting debug requests. Core compare error will not be generated and flags will not update. A CPU reset is needed to ensure the CPUs are again in lockstep and will also re-enable the CCM-R4.

8.4 CCM-R4 Control Registers

Table 8-3 lists the CCM-R4 registers. Each register begins on a 32-bit word boundary. The registers support 32-bit, 16-bit, and 8-bit accesses. The base address for the control registers is FFFF F600h.

Table 8-3. CCM-R4 Control Registers

Offset	Acronym	Register Description	Section
FFFF F600h	CCMSR	CCM-R4 Status Register	Section 8.4.1
FFFF F604h	CCMKEYR	CCM-R4 Key Register	Section 8.4.2

8.4.1 CCM-R4 Status Register (CCMSR)

The contents of this register should be interpreted in context of what test was selected. That is, in what mode the CCM is operating.

Figure 8-2. CCM-R4 Status Register (CCMSR) (offset = FFFF F600h)

31	17	16
Reserved		CPME
R-0		R/W1CP-0

15	9	8	7	2	1	0
Reserved		STC	Reserved		STET	STE
R-0		R-0	R-0		R-0	R-0

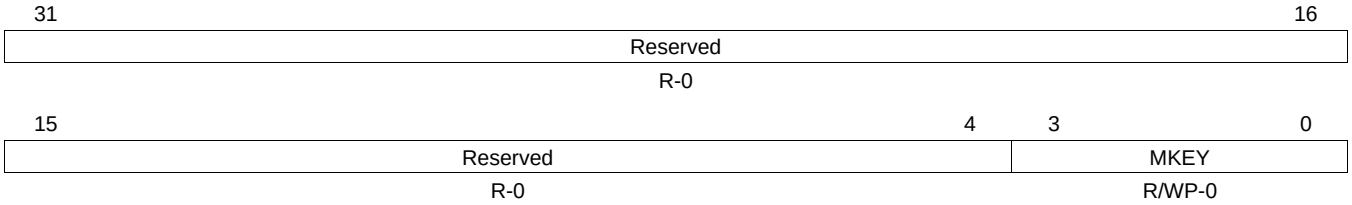
LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 8-4. CCM-R4 Status Register (CCMSR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	CMPE		Compare Error **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: CPU signals are identical. Write: Leaves the bit unchanged.
		1	Read: CPU signal compare mismatch. Write: Clears the bit.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	STC		Self-test Complete **Note:** This bit is always 0 when not in self-test mode. Once set, switching from self-test mode to other modes will clear this bit. **Read/Write in User and Privileged mode.**
		0	Read: Self-test on-going if self-test mode is entered. Write: Writes have no effect.
		1	Read: Self-test is complete. Write: Writes have no effect.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	STET		Self-test Error Type **Read/Write in User and Privileged mode.**
		0	Read: Self-test failed during Compare Match Test if STE = 1. Write: Writes have no effect.
		1	Read: Self-test failed during Compare Mismatch Test if STE = 1. Write: Writes have no effect.
0	STE		Self-test Error **Note:** This bit gets updated when the self-test is complete or an error is detected. **Read/Write in User and Privileged mode.**
		0	Read: Self-test passed. Write: Writes have no effect.
		1	Read: Self-test failed. Write: Writes have no effect.

8.4.2 CCM-R4 Key Register (CCMKEYR)

Figure 8-3. CCM-R4 Key Register (CCMKEYR) (offset = FFFF F604h)



LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privileged mode only; -*n* = value after reset

Table 8-5. CCM-R4 Key Register (CCMKEYR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MKEY		Mode Key **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Returns current value of the MKEY. Write: Lockstep mode.
		6h	Read: Returns current value of the MKEY. Write: Self-test mode.
		9h	Read: Returns current value of the MKEY. Write: Error Forcing mode.
		Fh	Read: Returns current value of the MKEY. Write: Self-test Error Forcing mode. **Note:** It is recommended to not write any other key combinations. Invalid keys will result in switching operation to lockstep mode.

Oscillator, PLL, and Clock Monitoring

This chapter describes the oscillator and PLL clock source paths for the device.

Topic **Page**

9.1 Introduction

This chapter provides an overview of the oscillator and PLL clock source paths for the device.

The oscillator macro will pass a signal driven into the OSCIN pin to clock source 0 that is the device default clock source on reset. When a crystal or resonator with appropriate load circuitry is connected to OSCIN and OSCOUT, the oscillator macro drives the crystal/resonator to generate the input waveform. In addition to being directly usable as clock source 0, the oscillator clock is the input to the PLL.

The oscillator frequency is continuously monitored by a dedicated clock detect circuit. If the frequency falls out of a fixed range, the clock detect switches the clock from the oscillator to an internally generated, free-running frequency (from the low power oscillator (LPO)).

The phase lock loop (PLL), a circuit in the microcontroller, is used to multiply the input frequency to some higher (device operation) frequency. This frequency synthesis is useful for generating higher frequencies than can be conveniently achieved with an external crystal or resonator. Additionally, the PLL allows the flexibility to be able to synthesize one of multiple frequency options from a given crystal or resonator.

Frequency modulation can be superimposed on the synthesized frequency. The modulation provides a means to reduce the impact of electromagnetic radiation from the device; this reduction in measured radiation can be useful in sensitive applications.

9.1.1 Features

The main features of the source clock path are:

- The oscillator may drive a crystal/resonator or be driven from an external source
- The clock detect provides continuous monitoring of the oscillator frequency and provides an automatic switch over to a free-running clock in case of oscillator failure.
- The FM-PLL module can be operated in either modulation or non-modulation mode.
- The phase-frequency detector assures lock to the fundamental reference frequency.
- $$f_{PLL} = \frac{f_{OSCIN}}{NR} \times \frac{NF}{OD \times R} \quad (1)$$
  - Configurable prescale divider (NR) for the input clock
  - Configurable multiplier (NF)
  - Configurable postscale dividers (OD, R)
- The PLL may be used with modulation enabled.
  - Configurable modulation frequency (NS)
  - Configurable modulation depth (NV)
- The slip control circuitry provides flexible response to a PLL failure (slip) including reset or automatic switch over to oscillator.

9.2 Quick Start

The purpose of this section is to provide an overview of how to configure the oscillator and PLL clock paths on power-up. More detailed descriptions are presented in later sections. Figure 9-1 shows the oscillator and PLL clock paths.

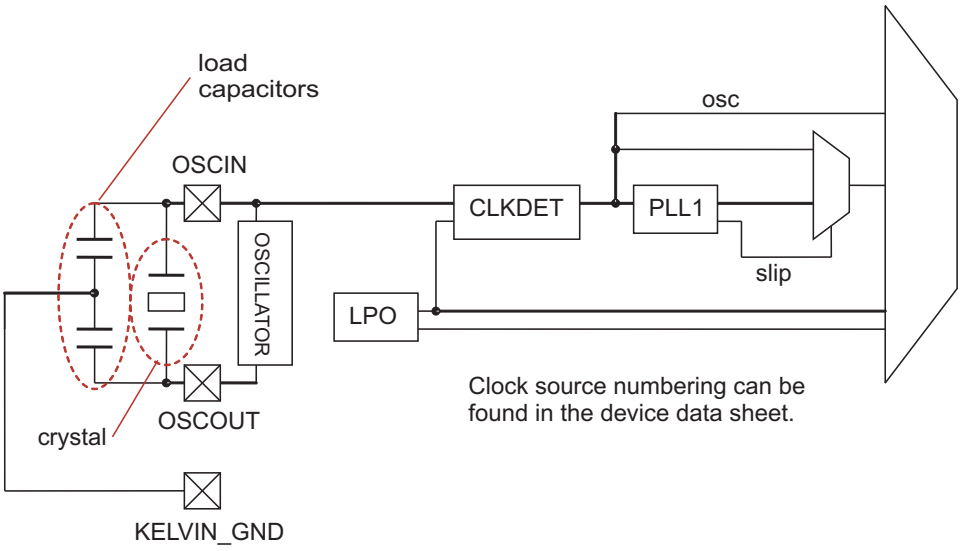
While power-on reset is asserted (low), the oscillator and low power oscillator (LPO) are enabled and start-up by default. After power-on reset is released to a high level, the clock detect circuit (CLKDET) begins to monitor the oscillator. If the oscillator is within a valid range, the oscillator becomes the default clock for the device as it exits reset; if the oscillator is not within a valid range, the clock detect selects the high-frequency low power oscillator as the default clock for the device.

The low power oscillator has a wide frequency range which also creates a large valid window for the clock detect; in order to refine the clock detect window, the low power oscillator can be trimmed. The initial trim value is stored in one-time programmable section of the flash memory, address 0xF008_01B4. Bits 31:16 of this word contain a 16 bit value that may be programmed into LPOMONCTL(15:0) in order to initialize the trim for both HF LPO and LF LPO. Software should read the initial trim values from flash and write them to the control register.

The PLL is disabled by default on power-up. The PLL control registers (PLLCTL1 and PLLCTL2) must be configured to set the desired output frequency. Then, the system PLL may be enabled (CLRCLKSR1OFF bit in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers). The PLL has a valid bit that indicates the PLL is locked (CLKSR1V bit in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers).

Prior to selecting the PLL clock as the source for a clock domain (for example, GCLK, HCLK, VCLKA1), the domain and modules on the domain must be configured to accept the new frequency. An example of a module that should be configured prior to selecting the PLL as clock source for GCLK and HCLK is the memory wrapper to insure that access times are maintained correctly.

Figure 9-1. Clock Path From Oscillator Through PLL To Device



9.3 Oscillator

The clock generation path through the PLL begins with the oscillator. The oscillator consists of three separate pads -- OSCIN, OSCOUT, and KELVIN_GND (see Figure 9-2).

The oscillator is responsible for two independent functions:

1. The oscillator is responsible for generating positive feedback in the external crystal/resonator with appropriate load and tank circuitry. At start-up, the oscillator amplifies random noise. The external circuitry acts like a band-pass and selects the crystal/resonator frequency to provide as positive feedback into the amplifier. The positive feedback increases the amplitude of the output waveform into the crystal/resonator (and the load circuitry), and the voltage waveform shows an envelope of increasing amplitude. The oscillator can drive a crystal frequency that is within the data sheet range $t_{c(OSC)}$.

   Looking at the input waveform into OSCIN, the voltage waveform is an AC-coupled, filtered version of the OSCOUT waveform. The band-pass functionality of the crystal/resonator removes distortion from the OSCOUT waveform, leaving a sinusoidal input waveform.

   NOTE: Vendor Validation of Resonators/Crystals

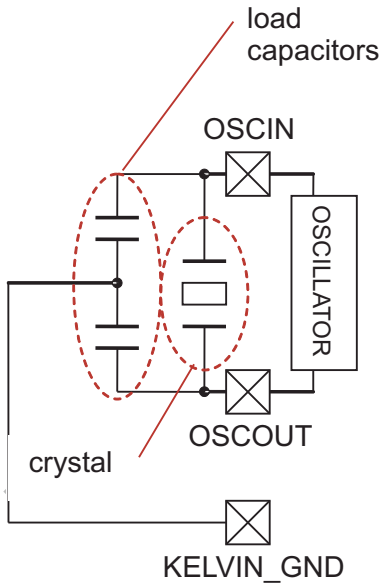
   The crystal is a very tight bandpass filter while a resonator is a somewhat wider bandpass. The load circuitry pulls the center frequency of the bandpass.

   Texas Instruments strongly encourages each customer to submit samples of the device to the resonator/crystal vendor for validation. The vendor is equipped to determine what load capacitances will best tune their resonator/crystal to the microcontroller device for optimum start-up and operation over temperature and voltage extremes. The vendor also factors in margins for variations in the microcontroller process.

2. The oscillator is also responsible for squaring-up the input waveform. This squaring-up converts the sinusoid into a square wave at the core logic levels. The input path limits the input frequency range as a low-pass filter with a cutoff frequency.

   The oscillator has a frequency range that is determined by the driving capability of external crystals/resonators (feedback path). If a clock is driven directly into the oscillator, then the feedback path is not relevant and the frequency range is determined solely by the forward path (that typically allows a higher frequency); the device can support inputs within the data sheet range $t_{c(OSC\_Sqr)}$.

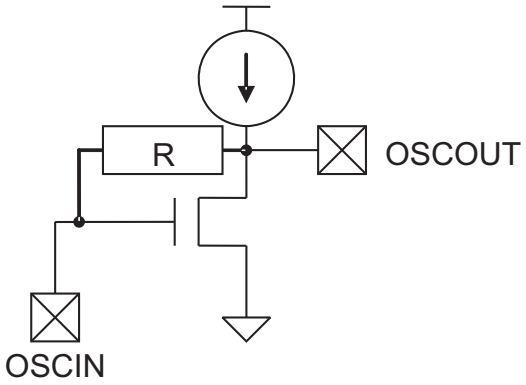
Figure 9-2. Clock Generation Path



9.3.1 Oscillator Implementation

The oscillator operates at 3.3V and uses a constant current source to drive current onto the OSCOUT node (see Figure 9-3). An internal transistor shunts the current (and current from the external circuitry) to GND. This current steering drives the voltage waveform on OSCOUT.

Figure 9-3. Oscillator Implementation



9.3.2 Oscillator Enable

The oscillator is enabled asynchronously when nPORRST is low.

The oscillator is enabled by clearing bit 0 in the Clock Source Disable Register (CSDIS) or setting bit 0 in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers. The bit sends a start signal to the oscillator. Bit 0 of CSDIS is cleared to 0 by default on a system or power-on reset so that the oscillator starts-up by default. After the oscillator swings at a high-enough amplitude to pass an input clock into the core domain and nPORRST is released, 1024 oscillator periods are counted before setting the CLKSR0V bit in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers. The oscillator generates clock source 0 in the global clock module (GCM).

9.3.3 Oscillator Disable

The clock sources (for example, OSC, PLL) are disabled by setting the appropriate bit in the Clock Source Disable Register (CSDIS) or setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET) of the System and Peripheral Control Registers. These bits *allow* the clock source to disable but do not force the behavior until the clock is no longer used as the source for a clock domain (for example, GCLK, VCLK, VCLK2, RTICLK). The CLKSR0V bit in the Clock Source Valid Status Register (CSVSTAT), of the System and Peripheral Control Registers, is cleared after clock disable is asserted (which occurs after all clock domains are stopped).

The oscillator disable signal places the oscillator into a low power state, disconnects the feedback (bias) resistor between OSCIN and OSCOUT, and OSCIN is grounded.

9.4 Low-Power Oscillator and Clock Detect (LPOCLKDET)

The Low-Power Oscillator (LPO) is comprised of two oscillators -- HF LPO and LF LPO -- in a single macro. The low-power oscillator and clock detect (LPOCLKDET) uses a relaxation oscillator to generate an internal clock whose frequency is NOT tightly controlled. This frequency is used to monitor the oscillator input frequency and is also available as an independent clock source in the GCM.

The LPO produces two frequencies:

- High-frequency low-power oscillator (HF LPO) with a nominal frequency of 9.6MHz; the HF LPO generates clock source 5 in the GCM. Refer to the device datasheet for range values.
- Low-frequency low-power oscillator (LF LPO) with a nominal frequency of 85KHz; the LF LPO generates clock source 4 in the GCM. Refer to the device datasheet for range values.

A single current source drives current onto a capacitor; when the voltage on the capacitor exceeds some threshold, the clock toggles. The LPO uses a single current source and the two different comparators to generate the HF LPO and LF LPO frequencies. The LPO is controlled by 4 different bit fields -- CSDIS(5:4), HFTRIM, LFTRIM, and BIASEN.

- CSDIS.5 enables/disables the comparator that generates HF LPO.
- CSDIS.4 enables/disables the comparator that generates LF LPO.
- The HFTRIM and LFTRIM bit fields vary the current into the comparator to independently trim the HF LPO and LF LPO frequencies (see Section 2.5.1.31).
- BIAS ENABLE enables/disables the current source that drives the LPO.

9.4.1 Clock Detect

The LPO HF clock frequency is typically near 9.6MHz. Refer to the device datasheet for range values. The clock detect establishes a window for the oscillator by:

OSCIN > HF LPO_{min} / 4	OSCIN / 4 < HF LPO_{max}
OSCIN > 5.5[MHz] / 4 = 1.375[MHz]	OSCIN < 4 × 19.5 = 78[MHz]

The clock detect circuit works by checking for a rising edge on one clock (oscillator or HF LPO) between rising edges of the other clock. The result is that in addition to flagging incorrect, repeating frequencies, the circuit also fails due to transient conditions.

NOTE: Clock Detection of Oscillator MUST be Disabled Before Disabling HF LPO

The HF LPO frequency is the comparison frequency for the oscillator. The clock detection must be disabled prior to disabling the HF LPO frequency.

If the clock detection is NOT disabled prior to disabling the HF LPO, the clock detect circuitry will fail the oscillator as too fast (compared to the non-existent HF LPO). The clock detect circuitry will switch to the non-existent clock, leaving the device without a valid clock.

9.4.2 Behavior on Oscillator Failure

If the oscillator frequency fails, the clock detects supplies:

- the HF LPO clock to GCM clock source 0 instead of the oscillator
- the HF LPO clock to GCM clock source 1 instead of the PLL

The HF LPO signal will be available as three different clock sources:

- GCM clock source 0 (replacing the oscillator)
- GCM clock source 1 (replacing the PLL)
- GCM clock source 5 as HF LPO

The automatic switch-over from oscillator to HF LPO allows the application to execute at a reduced frequency and respond to a problem with the external crystal/resonator. During and after an oscillator failure, the oscillator CLKSRnV bit in the Clock Source Valid Status Register (CSVSTAT), of the System and Peripheral Control Registers, is set along with the OSCFAIL flag in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers.

It is useful to explicitly change the GHVSRC register, defining the current clock source for GCLK/HCLK/VCLK domains, to the HF LPO after an oscillator failure.

When reset on oscillator failure is set, PLLCTL1.23 (ROF), the device responds to an oscillator failure by generating a device reset.

9.4.3 Recovery from Oscillator Failure

If the oscillator fails, the clock detect switches the HF LPO frequency onto the oscillator source into the GCM. The OSCFAIL flag in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers, is also set.

The oscillator may be re-enabled (though if the failure was caused by a hard-fault, the re-enable will fail) through the following procedure:

1. Switch all clock domains from the oscillator to the HF LPO (for example, GHVSRC uses HF LPO, VCLKA1 uses HF LPO or VCLK, and so on).
2. If the PLL is used, disable the PLL by setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET) of the System and Peripheral Control Registers.
3. Disable the oscillator by setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET). This action resets the clock detect and allows the oscillator to propagate through GCM clock source 0.
4. Re-enable the oscillator by setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers.
5. Clear the OSCFAIL flag in the Global Status Register (GLBSTAT) by writing a 1 to the bit. The PLL slip bits may also be set on an oscillator failure. These can also be cleared.
6. Switch the clock domains back to the oscillator.
7. Re-enable the PLL by setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR).

NOTE: Clock Re-Enable Procedure Will Fail If Caused by a Hard Failure

Although it is possible to re-enable the oscillator after a failure, if the oscillator failure was caused by a hard fault (for example, disconnected crystal/resonator terminal), the re-enable process will fail.

9.4.4 LPOCLKDET Enable

The LPO is enabled by default while nPORRST is low. During this time, the current source initializes, holding the relaxation oscillator in reset until initialized. After the current source releases the HF LPO and the LF LPO, these clock frequencies slew to their final frequencies; the final frequency may be achieved while nPORRST is active or after its release. After, nPORRST is released, the HF LPO Valid signal is set 32 HF LPO clock cycles later.

The clock detect is enabled once the oscillator and HF LPO are valid. Because an oscillator failure could occur from reset, the clock detect logic must provide an override path. If the HF LPO is valid and the oscillator is not valid, the clock detect circuitry will become active (overriding the oscillator invalid signal) after 16K LF LPO cycles (about 200 ms).

9.4.5 LPOCLKDET Disable

9.4.5.1 Disable Clock Detect

It is possible to disable the clock detect circuitry. For protection, this clock detect disable employs a 2-bit key:

- RANGE DET ENA SSET (CLKTEST.24) must be set to 1
- RANGE DET CTRL (CLKTEST.25) must be cleared to 0

In this case, the LPO HF and LF clocks are still active but the clock detect circuitry is disabled. The clock detect unconditionally switches GCM_CLK_SRC(0) back to the oscillator so care should be taken to insure that the oscillator is good before disabling the clock detect circuitry.

9.4.5.2 Disable LPO HF and LF Clocks

The LPO may be disabled by holding the relaxation oscillator clocks (HF and LF) in reset. The clock detect must be disabled, and any clock domains using either HF or LF clocks must be switched to a different clock source. The LPO HF clock is reset by setting CSDIS.5; CSDISSET.5 is an easy way to set specific bits without disturbing the rest of the register. The HF LPO clock disables several HF LPO cycles after CSDIS is set.

Similarly, the LPO LF clock is reset by setting CSDIS.4, and in a similar way CSDISSET.4 can set the specific CSDIS register bit without using a read-modify-write construction. The LF LPO disables several LF LPO cycles after CSDIS is set.

Restarting the LPO clocks from this condition is fast and is known as a warm re-start. The CSDISCLR register allows the user to clear CSDIS bits without using a read-modify-write code-construct.

9.4.5.3 Disable LPO Current Bias

The LPO current source may be disabled after the clock detect is disabled and HF and LF clock sources are disabled. Turning off this current source places the LPOCLKDET into its lowest power configuration. The bias may be disabled by clearing the BIAS ENABLE bit.

Restarting the LPO when the bias current has been disabled requires the current source to initialize first and is, therefore slower than a warm re-start; re-enabling the LPO from this condition is known as a warm re-start (similar to what happens during nPORRST active).

9.4.6 Trimming the HF LPO Oscillator

The HF LPO range varies considerably around 9.6MHz from device to device. In order to provide tighter monitoring of the crystal/resonator, it is useful to trim the oscillator. During device test, a trim value is written into the one-time programmable section of the flash memory (OTP), address 0xF008_01B4. Bits 31:16 of this OTP word contain a 16-bit value that may be programmed into LPOMONCTL[15:0] in order to initialize the trim for both HF LPO and LF LPO.

When trimming the HF LPO, it is recommended to step the trim value so as not to make a large change to any TRIM setting.

After the initial trim, further trimming may be done in LPOMONCTL, using the dual clock compare module (see Chapter 10) in order to determine the resultant frequency. This module allows for comparison of two clock frequencies. Once the HF LPO is determined to be in-range with the initial HFTRIM setting from the OTP, the crystal oscillator may be used as a reference against which the HF LPO and LF LPO may be further adjusted.

9.5 PLL

The following bit fields from PLLCTL1 and PLLCTL2 configure the PLL:

- REFCLKDIV[5:0]
- PLLMUL[15:0]
- ODPLL[2:0]
- PLLDIV[4:0]
- SPR_AMOUNT[8:0]
- SPREADINGRATE[8:0]
- FMENA

The PLL is responsible for synthesizing an output frequency from the input clock (from the oscillator); Figure 9-4 shows a simple block diagram of the PLL. The FM-PLL divides the reference input for a lower frequency input into the PLL ($f_{INTCLK} = f_{CLKIN}/NR$). The PLL multiplies this internal frequency by NF to get the VCO output clock frequency ($f_{Output\ CLK} = f_{INTCLK} \times NF$). The PLL output is subsequently divided by two prescale values (OD and R). The value of OD is an integer from 1-8 and R is an integer from 1–32. This output clock, PLL CLK, sources GCM clock source 1. Valid frequencies are shown in Table 9-1 while Table 9-2 shows how that encoding is generated from the PLL bit fields.

[$f_{(post\_ODCLK)}$ and $f_{(GCLK)}$ are data sheet parameters.]

Figure 9-4. Operation of the FM-PLL Module

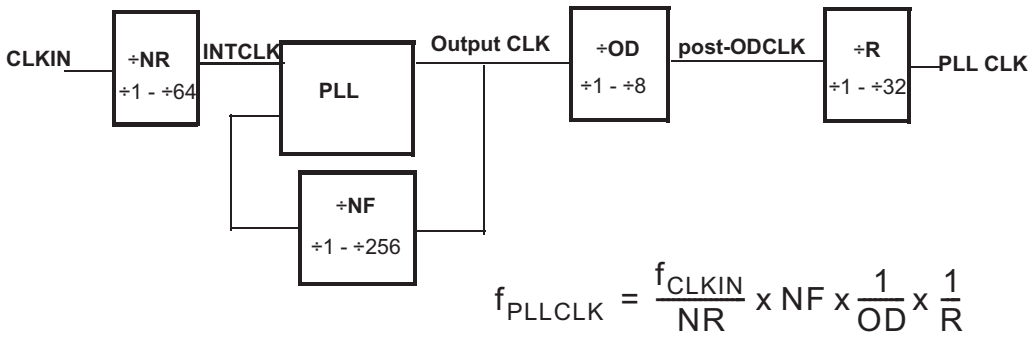


Table 9-1. Valid Frequency Ranges for PLL

	Frequency Limit
f_{CLKIN}	$f_{(OSC\_Sqr)}$
f_{INTCLK}	1MHz - $f_{(OSC\_Sqr)}$
$f_{Output\ CLK}$	150MHz - 550MHz
$f_{post\text{-}ODCLK}$	$f_{(post\_ODCLK)}$
$f_{PLL\ CLK}$	$f_{(GCLK)}$

Table 9-2. PLL Value Encoding

	PLL
NR	$NR = REFCLKDIV[5...0]+1$ (2)
NF	Non-modulated: $NF = \frac{(PLLMUL[15...0] + 256)}{256}$ (3) Modulated: $NF = \frac{(PLLMUL[15...0] + MULMOD[8...0] + 256)}{256}$ (4)
NV	$NV = \frac{(SPR\_AMOUNT[8...0]+1)}{2048}$ (5)
NS	$NS = SPRRATE[8...0]+1$ (6)
OD	$OD = ODPLL[2...0]+1$ (7)

NOTE: ODPLL change should occur prior to enabling asynchronous clock domains

Since changing the ODPLL bit-field causes the PLL CLK to be gated, these changes to ODPLL should be completed before configuring a clock domain for an asynchronous clock source. Some clock domains (RTICLK, VCLK2) require a frequency relationship to the VCLK.

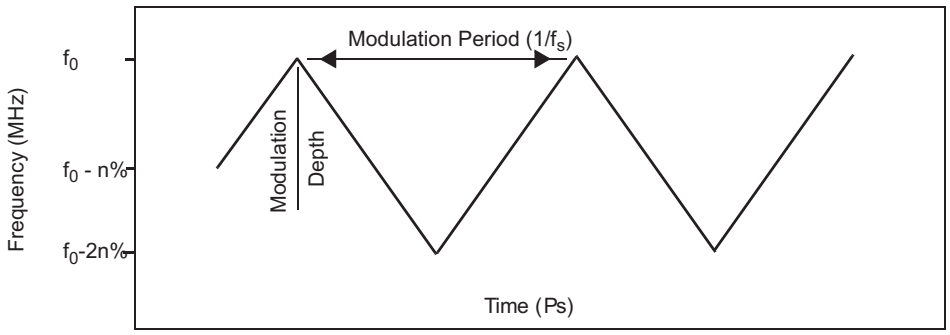
$$f_{VCLK} \geq 3 \times \frac{f_{RTISRC}}{RTIDIV}$$

If the PLL is clocking VCLK and it is stopped for some cycles, then the frequency relationship is temporarily violated.

Many asynchronous domains require frequency relationships between VCLK and the asynchronous domain. Therefore, if the PLL clock is the source for GCLK, HCLK and VCLK, then the gating produces a short-term change in the PLL clock frequency (and hence, also the VCLK frequency). As such, this frequency change could violate the requirements for an asynchronous clock domain.

9.5.1 Modulation

Optionally, the frequency can be modulated, that is, a controlled jitter is introduced onto the baseline frequency of the PLL. This modulation mechanism is not shown in Figure 9-4. When the PLL is used in the modulating mode, the programmable modulation block varies the PLL frequency from the baseline frequency ($f_{baseline}$ = (f_{CLKIN}/NR) × NF/(OD × R)) to $f_{baseline}$ × (1 - 2 × Depth) in a period defined by 1/f_s; the modulation waveform is triangular and should be enabled after lock.



The modulation is digital and the spreading profile is triangular, down-spread which implies:

- the modulation waveform is composed of a series of frequency steps.
- the modulation frequency and modulation depth are both well controlled due to their digital character.
- the average frequency during modulation is lower than the average frequency prior to enabling modulation. The depth of modulation, however, sets the new average frequency.
- the modulation frequency must be selected slower than the loop bandwidth. From a practical perspective, NS should be near 20.

The modulation fields have a simple geometric meaning:

- the modulation step size is:

$$\frac{NV}{NF} \times f_{OutputCLK} \tag{8}$$

- the number of steps per modulation period is 2 × NS
- the modulation depth is given by:

$$\Delta f = \frac{NS}{2} \times \frac{NV}{NF} \times f_{OutputCLK}$$

$$Depth[\%] = \frac{NS}{2} \times \frac{NV}{NF} \tag{9}$$

- the modulation frequency is:

$$T_{\text{mod}} = \frac{f_{OSC}}{2 \times NR \times NS} \tag{10}$$

- MULMOD minimizes frequency offset when programmed as:

$$\frac{(SPR\_AMOUNT[8\ldots0]+1)(SPRRATE[8\ldots0]+1)}{16} \tag{11}$$

NOTE: Modulation should be enabled after Lock

Enable modulation after the lock is completed.

9.5.2 PLL Output Control

The outputs from the PLL are the output clock, slip signals, and VALID.

- RFSLIP -- the RFSLIP signal indicates that the Output CLK is running *too fast* relative to INTCLK and sets a RFSLIP status flag in the Global Status Register (GLBSTAT) register, of the System and Peripheral Control Registers, if the slip signal is active during normal PLL operation; the RFSLIP flag is masked off while the PLL is not active and during the PLL's lock period.
- FBSLIP -- the FBSLIP signal indicates that the Output CLK is running *too slow* relative to INTCLK and sets a FBSLIP status flag in the Global Status Register (GLBSTAT) register, of the System and Peripheral Control Registers, if the slip signal is active during normal PLL operation; the FBSLIP flag is masked off while the PLL is not active and during the PLL's lock period.
- PLL Slip -- Logical-OR of the two PLL slip signals. Typically this signal is used to generate a consolidated slip signal to the device (for example, error logic or exception generation). Also used to gate VALID.

NOTE: Clearing Slip Bits

In order to clear any of the slip bits, it is necessary to disable the PLL first.

- VALID -- is driven based upon whether the output clock, PLL CLK, is gated or not. However, the VALID signal is dependent upon the PLL Slip signals so that VALID cannot be set if either slip signal is active.
- PLL Clock -- The PLL output clock runs at the programmed frequency. When enabled, it takes some time to acquire the programmed frequency (see Section 9.5.2.1). Similarly, the disable has some timing/constraints (see Section 9.5.2.2).

9.5.2.1 PLL Enable

After setting the PLL control registers, the clock source is enabled by clearing the appropriate bit in the Clock Source Disable Register (CSDIS) or setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers. The bit sends a signal to the PLL that starts the process of enabling the PLL.

1. The PLL checks to make sure that the oscillator is ON. If not, it turns the oscillator ON.
2. The PLL begins a locking process in which the PLL slews from a starting frequency point to the programmed frequency. During this lock period, the PLL slip signals are typically active, and the PLL masks off the signals during this phase. The lock phase takes the following length of time:

Parameter	Value
Lock	$T_{Lock} = (512 \times T_{OSCIN}) + (1024 \times NR \times T_{OSCIN})$
Enable clocks after lock	$T_{Enable} = 6 \times T_{OSCIN}$

3. After the lock phase is complete (when lock counters expire), the PLL releases the slip signals to the system.
4. Then, after the slip signals are released and a delay to enable the clocks, the clock is released to the system and the appropriate CLKSRnV bit for the PLL is set in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers.

9.5.2.2 PLL Disable

The clock sources (for example, OSC, PLL) are disabled by setting the appropriate bit in the Clock Source Disable Register (CSDIS) or setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET) of the System and Peripheral Control Registers. These bits *allow* the clock to disable but do not force the behavior until the clock is no longer used as the source for a clock domain (for example, GCLK, VCLK, VCLK2, RTICLK).

The PLL receives a signal to disable after the clock is no longer used by any clock domain. Within the PLL, the clock is disabled and the appropriate CLKSRnV bit for the PLL in the Clock Source Valid Status Register (CSVSTAT), of the System and Peripheral Control Registers, becomes inactive. Then the PLL is placed into a low power state after the following length of time: $T_{Enable} = 150 \times T_{OSCIN}$

9.5.2.3 OD-divider Change

The PLL gates the clock if the ODPLL bit-field is changed while the PLL is active. The output clock from the PLL is gated for 3 or 12 OSCIN clock cycles. As the post-ODCLK is gated in the low phase, the output clock to the device -- PLL CLK -- may be gated in a high or low phase though the transition is always glitchless: $T_{ODPLL} = 3 \times T_{OSCIN}$

NOTE: ODPLL change should occur prior to enabling asynchronous clock domains

Since changing the ODPLL bit-field causes the PLL CLK to be gated, these changes to ODPLL should be completed before configuring a clock domain for an asynchronous clock source. Some clock domains (RTICLK, VCLK2) require a frequency relationship to the VCLK.

$$f_{VCLK} \geq 3 \times \frac{f_{RTISRC}}{RTIDIV}$$

If the PLL is clocking VCLK and it is stopped for some cycles, then the frequency relationship is temporarily violated.

Many asynchronous domains require frequency relationships between VCLK and the asynchronous domain. Therefore, if the PLL clock is the source for GCLK, HCLK and VCLK, then the gating produces a short-term change in the PLL clock frequency (and hence, also the VCLK frequency). As such, this frequency change could violate the requirements for an asynchronous clock domain.

9.5.2.4 Changing the PLL Operating Point While the PLL is Active

Once the valid bit (CLKSRnV bit in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers) is set, software may change values to the PLL. If the change of values results in a small percentage change to the VCO frequency ($\Delta f_{OutputCLK} < 0.1 \times f_{OutputCLK}$), then these changes can be done on-the-fly. In this mode, the values are updated into the PLL synchronously, and the PLL re-locks to the new value without gating the clocks or the slip bits. If the operating point change is too large, then the slip bits will be set.

Conversely, if the changes to the VCO frequency are large, then the PLL should be disabled prior to changing the values. Typically, any change to the REFCLKDIV field or large changes to the PLLMUL field in the PLL Control Register 1 (PLLCTL1) of the System and Peripheral Control Registers requires a complete disable-and-relock strategy.

9.5.2.5 Summary of PLL Timings

In addition to controlling the lock period and disabling the clock during an ODPLL change, the PLL also generates reset delays. When power-on reset is released (nPORRST 0 --> 1), that release is delayed by 1024 OSCIN cycles so that it is released at the same time that the oscillator valid is asserted. The system reset release is delayed by an additional 8 oscillator clock cycles.

Table 9-3. Summary of PLL Timings

Parameter	Value
nPORRST delay	$T_{nPORRST} = 1024 \times T_{OSCIN}$
nRST delay	$T_{nRST} = 1032 \times T_{OSCIN}$
OSC valid	$T_{OSCVALID} = 1024 \times T_{OSCIN}$
Lock	$T_{Lock} = (512 \times T_{OSCIN}) + (1024 \times NR \times T_{OSCIN})$
Enable clocks after lock	$T_{Enable} = 6 \times T_{OSCIN}$
Disable clocks after lock	$T_{Enable} = 150 \times T_{OSCIN}$
Change ODPLL	$T_{ODPLL} = 3 \times T_{OSCIN}$

9.5.3 Behavior on PLL Fail

The PLL allows flexible response to a PLL failure (slip). Like the oscillator, the PLL clock is configured by default to automatically switch-over to the oscillator in case of a PLL slip. (In this case, the oscillator sources GCM clock source 1 as well as GCM clock source 0. Also, if the oscillator fails, LPO HF is sourced to both GCM clock sources 0 and 1.)

A slip after the PLL has locked and while it is active is an indication of a PLL failure. The PLL provides slip-filtering which enhances the flexibility of the PLL's response to failure. The slip-filtering circuit samples the slip based on HF LPO. The filter defines the number of consecutive HF LPO cycles for which the slip signal must be active before the slip is recognized. This slip is latched in the RFSLIP and FBSLIP status flags in the Global Status Register (GLBSTAT) register of the System and Peripheral Control Registers.

The PLL may enable/disable the automatic switch over as well as the error signaling; if the error signaling is enabled, a PLL slip may be configured to generate a reset. The automatic switch-over and suppression of the error signals are controlled by the bypass on slip bit field -- BPOS[1:0] (PLLCTL1.(30:29)). When BPOS[1:0] is disabled (BPOS[1:0] = 10b):

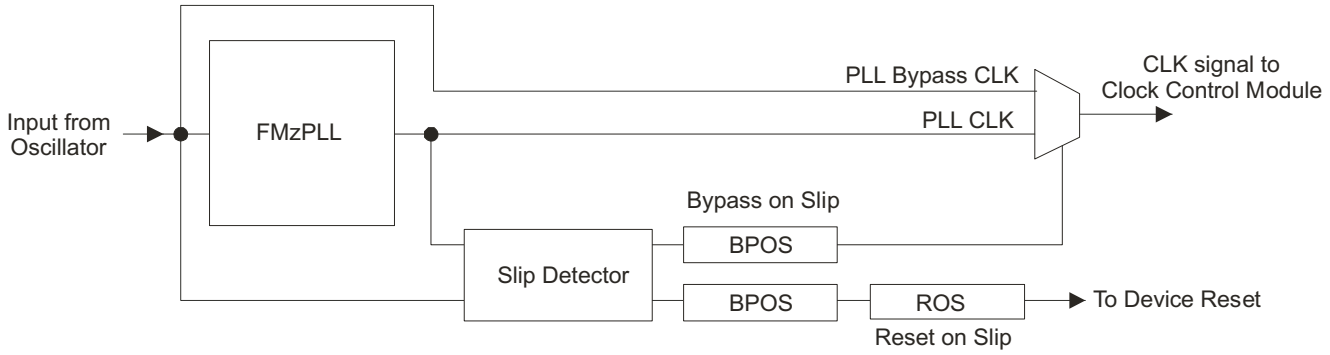
- automatic response to the PLL slip is prevented
- ESM/exception is NOT generated
- reset on slip is not generated regardless of the state of the ROS bit
- status bits are set on a PLL slip independent of BPOS[1:0]

When BPOS[1:0] is enabled (BPOS[1:0] = 00b OR 01b OR 11b):

- PLL slip causes the clock source into GCM clock source 1 to shift from the PLL to the oscillator
- ESM/exception is generated
- reset on slip is generated if ROS is set

The effect of BPOS[1:0] on the system is shown in Figure 9-5.

Figure 9-5. PLL Slip Detection and Reset/Bypass Block Diagram



9.5.4 Recovery from a PLL Failure

If the PLL fails, the PLL slip causes the valid flag to be locked and causes the clock source into GCM clock source 1 to shift from the PLL to the oscillator. This slip is latched in the RFSLIP and FBSLIP status flags in the Global Status Register (GLBSTAT) register of the System and Peripheral Control Registers.

The PLL may be re-enabled (though if the failure was caused by a hard-fault, the re-enable will fail) through the following procedure:

1. Switch all clock domains from the PLL to the oscillator (for example, GHVSRC uses oscillator, VCLKA1 uses oscillator or VCLK, and so on).
2. Disable the PLL by setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET) of the System and Peripheral Control Registers. This action disables the PLL and causes the slip signal to no longer be driven. VALID is not released until the slip is cleared.
3. Clear the RFSLIP or FBSLIP flags in the Global Status Register (GLBSTAT) register, of the System and Peripheral Control Registers, by writing a 1 to the bits. After this step, the valid flag is unlocked and cleared if it was previously set.
4. Re-enable the PLL by setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers.
5. Switch the clock domains back to the PLL.

9.5.5 PLL Modulation Depth Measurement

The PLL contains a circuit for estimating the depth of the modulation. The circuit counts clock edges over a fixed window of the modulation waveform (SSW_CAPTURE_COUNT in SSWPLL2) and clock edges over the entire waveform (SSW_CLKOUT_COUNT in SSWPLL3). The capture ends after a pre-determined number of clock edges in SSW_CLKOUT_COUNTER as set in TAP_COUNTER_DIS. There are 2 × NR windows per modulation waveform. The procedure for estimating the modulation depth is:

1. While GCLK is sourced by the oscillator and the PLL is enabled with modulation, configure SSWPLL1 as follows:
   a. CAPTURE_WINDOW_INDEX is set equal to NR.
   b. COUNTER_RESET is set.
   c. TAP_COUNTER_DIS is set to disable the measurement after SSW_CLKOUT_COUNT captures this number of clocks. The measurement is disabled after the set tap is set AND the modulation cycle ends.
   d. Ensure that EXT_COUNTER_EN is cleared.
2. Ensure that both SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT are cleared (by the COUNTER_RESET).
3. Set COUNTER_EN and clear COUNTER_RESET. This step releases the reset and enables the counter to begin counting.
4. After a wait loop, poll for COUNTER_READ_READY to set. After the bit is set, read SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT.
5. Compute the modulation depth as:

$$Depth = abs\left(1 - \frac{2 \times NR \times SSW\_CAPTURE\_COUNT}{SSW\_CLKOUT\_COUNT}\right) \quad (12)$$

9.5.6 PLL Frequency Measurement Circuit

The same circuit that is used to measure modulation depth is also available to measure the average frequency of the PLL. In this mode, the PLL output (before the R-divider) is captured in SSW_CLKOUT_COUNT while the oscillator is captured in SSW_CAPTURE_COUNT. The procedure for using the PLL frequency measurement circuit is:

1. While the PLL is enabled, set EXT_COUNTER_EN.
2. Set COUNTER_EN. This bit clears both SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT and then immediately enables for counting.
3. Wait for some software delay loop.
4. Clear COUNTER_EN. Wait for COUNTER_READ_READY to set. Read both SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT and compute the ratio of PLL multiplication as:

$$\frac{NF}{NR \times OD} = \frac{SSW\_CLKOUT\_COUNT}{SSW\_CAPTURE\_COUNT} \quad (13)$$

5. Note that CAPTURE_WINDOW_INDEX, COUNTER_RESET, TAP_COUNTER_DIS are not used in this procedure

9.6 PLL Control Registers

The clock module has two control registers (PLLCTL1 and PLLCTL2) located within the System and Peripheral Control Registers, plus it has four bits located in other System and Peripheral Control Registers.

The FM-PLL is off at power-on. The clock source is enabled by clearing the appropriate bit in the Clock Source Disable Register (CSDIS) or setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers. [CSDISCLR and Clock Source Disable Set Register (CSDISSET) also enable/disable the PLL and oscillator (and other clock sources).]

The LPOCLKDET module generates the OSCFAIL flag in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers, if a problem with the reference oscillator is detected. The slip signals are also registered in the RFSLIP and FBSLIP status flags in the Global Status Register (GLBSTAT) register, of the System and Peripheral Control Registers, in order to indicate the source of a clock failure.

The appropriate CLKSRnV bit for the PLL is set in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers.

The following sections describe the PLL registers used in the system module. These registers support 8-, 16-, and 32-bit write accesses. The reset values for these registers are configured so that an input frequency in the range from 5MHz to 20MHz generates a valid clock.

Table 9-4. PLL Module Registers

Address	Acronym	Register Description	Section
FFFF FF30h	CSDIS	Clock Source Disable Register	Section 2.5.1.10
FFFF FF34h	CSDISSET	Clock Source Disable Set Register	Section 2.5.1.11
FFFF FF38h	CSDISCLR	Clock Source Disable Clear Register	Section 2.5.1.12
FFFF FF54h	CSVSTAT	Clock Source Valid Status Register	Section 2.5.1.19
FFFF FF70h	PLLCTL1	PLL Control 1 Register	Section 2.5.1.26
FFFF FF74h	PLLCTL2	PLL Control 2 Register	Section 2.5.1.27
FFFF FFA0h	GPREG1	General Purpose Register	Section 2.5.1.35
FFFF FFECh	GLBSTAT	Global Status Register	Section 2.5.1.51
FFFF E170h	CLKSLIP	Clock Slip Register	Section 2.5.2.2
FFFF FF24h	SSWPLL1	PLL Modulation Depth Measurement Control Register	Section 9.6.1
FFFF FF28h	SSWPLL2	SSW PLL BIST Control Register 2	Section 9.6.2
FFFF FF2Ch	SSWPLL3	SSW PLL BIST Control Register 3	Section 9.6.3

Table 9-5. LPOCLKDET Module Registers

Address	Acronym	Register Description	Section
FFFF FF88h	LPOMONCTL	LPO/Clock Monitor Control Register	Section 2.5.1.31
FFFF FF8Ch	CLKTEST	Clock Test Register	Section 2.5.1.32

9.6.1 PLL Modulation Depth Measurement Control Register (SSWPLL1)

Figure 9-6 illustrates this register and Table 9-6 provides the bit descriptions.

Figure 9-6. SSW PLL BIST Control Register 1 (SSWPLL1) (offset = 24h)

31	16
Reserved	
R-0	

15	8
CAPTURE_WINDOW_INDEX	
R/W-0	

7	6	5	4	3	1	0
Reserved	COUNTER_ READ_ READY	COUNTER_ RESET	COUNTER_ EN	TAP_COUNTER_DIS		EXT_COUNTER_ EN
R-0	R-0	R/W-1	R/W-0	R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 9-6. SSW PLL BIST Control Register 1 (SSWPLL1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	CAPTURE_WINDOW_INDEX	0-FFh	The capture counter present in the PLL wrapper will count the PLL clock edges when the current modulation phase capture window value is equal to these bits. Should be set equal to NR.
7	Reserved	0	Reads return 0. Writes have no effect.
6	COUNTER_READ_READY		Counter read ready. Indicates that SSW_CAPTURE_COUNT (SSWPLL2) and SSW_CLKOUT_COUNT (SSWPLL3) can be read.
		0	Counter registers in SSWPLL2 and SSWPLL3 are not ready to read.
		1	Counter registers in SSWPLL2 and SSWPLL3 are ready to read.
5	COUNTER_RESET		Counter reset. If EXT_COUNTER_EN = 0, COUNTER_RESET resets SSW_CAPTURE_COUNT (SSWPLL2) and SSW_CLKOUT_COUNT (SSWPLL3). If EXT_COUNTER_EN = 1, this bit is ignored.
		0	No impact to counters.
		1	If EXT_COUNTER_EN = 0, then counters SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT will be held in the reset state. If EXT_COUNTER_EN = 1, this bit is ignored by the PLL wrapper.
4	COUNTER_EN		Counter enable. If EXT_COUNTER_EN = 0, COUNTER_EN initializes the modulation depth measurement. (In this mode, the disable is set to occur automatically.) If EXT_COUNTER_EN = 1, the counters are enabled/disabled with COUNTER_EN.
		0	If EXT_COUNTER_EN = 0, COUNTER_EN = 0 indicates that the counters are inactive. If EXT_COUNTER_EN = 1, COUNTER_EN = 0 disables the counters.
		1	If EXT_COUNTER_EN = 0, COUNTER_EN = 0 indicates that the counters are still active. If EXT_COUNTER_EN = 1, COUNTER_EN = 0 enables the counters.

Table 9-6. SSW PLL BIST Control Register 1 (SSWPLL1) Field Descriptions (continued)

Bit	Field	Value	Description
3-1	TAP_COUNTER_DIS		The value in this register is used to program a particular bit in CLKOUT counter. When that particular bit in CLKOUT counter becomes 1, then both the CLKOUT counter and the CAPTURE counter will stop counting when EXT_COUNTER_EN = 0. When EXT_COUNTER_EN = 1, this bit field is not used.
		0	Bit 16 of CLKOUT counter is selected. When this bit is set and the modulation period finishes, the counters are disabled and READ_READY_FLAG is set.
		1h	Bit 18 of CLKOUT counter is selected.
		2h	Bit 20 of CLKOUT counter is selected.
		3h	Bit 22 of CLKOUT counter is selected.
		4h	Bit 24 of CLKOUT counter is selected.
		5h	Bit 26 of CLKOUT counter is selected.
		6h	Bit 28 of CLKOUT counter is selected.
		7h	Bit 30 of CLKOUT counter is selected.
0	EXT_COUNTER_EN		Measurement mode.
		0	Modulation Depth Measurement mode.
		1	Frequency Measurement mode.

9.6.2 SSW PLL BIST Control Register 2 (SSWPLL2)

This is an observation register used to log counter value for the capture counter inside the PLL wrapper. The SSWPLL2 register is shown in Figure 9-7 and described in Table 9-7.

Figure 9-7. SSW PLL BIST Control Register 2 (SSWPLL2) (offset = 28h)

31	0
SSW_CAPTURE_COUNT	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 9-7. SSW PLL BIST Control Register 2 (SSWPLL2) Field Descriptions

Bit	Field	Description
31-0	SSW_CAPTURE_COUNT	Capture count. This register returns the value of the capture count. When EXT_COUNTER_EN = 0, this counter increments within a fixed modulation window. When EXT_COUNTER_EN = 1, this counter increments based upon the oscillator.

9.6.3 SSW PLL BIST Control Register 3 (SSWPLL3)

This is observation register used to log counter value for CLKOUT counter inside PLL wrapper. The SSWPLL3 register is shown in Figure 9-8 and described in Table 9-8.

Figure 9-8. SSW PLL BIST Control Register 3 (SSWPLL3) (offset = 2Ch)

31	0
SSW_CLKOUT_COUNT	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 9-8. SSW PLL BIST Control Register 3 (SSWPLL3) Field Descriptions

Bit	Field	Description
31-0	SSW_CLKOUT_COUNT	Value of CLKout count register. This counter increments based upon the PLL output (prior to the R-divider).

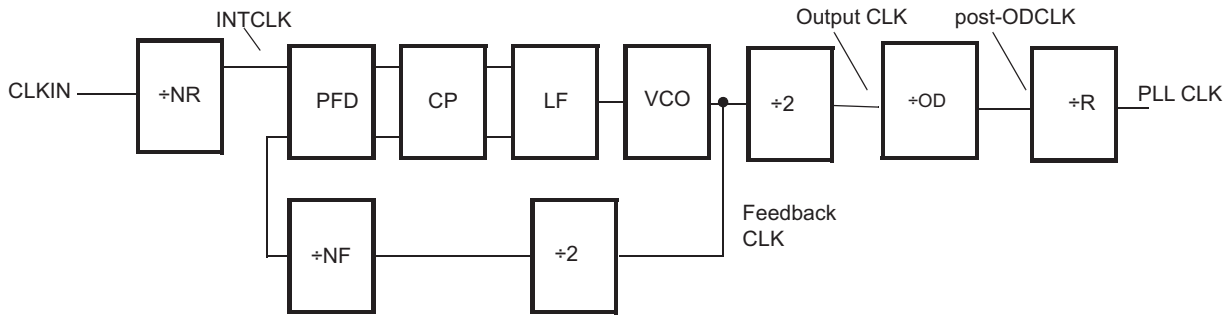
9.7 Phase-Locked Loop Theory of Operation

The PLL block consists of six logical sub-blocks:

- Phase-Frequency Detector (PFD)
- Charge Pump (CP)
- Loop Filter (LF)
- Voltage-Controlled Oscillator (VCO)
- Frequency Modulation
- Slip Detector

Figure 9-9 illustrates the sub-blocks in a basic PLL circuit. The VCO adjusts its frequency until the two signals into the PFD have the same phase and frequency. The feedback path (from VCO to PFD) divides the frequency of the feedback signal by 2 × NF; this feedback divider requires the VCO to generate a frequency 2 × NF times greater than the internal frequency (OSCIN/NR). In the forward path (from VCO to PLL CLK), the /2 block creates a clean duty cycle.

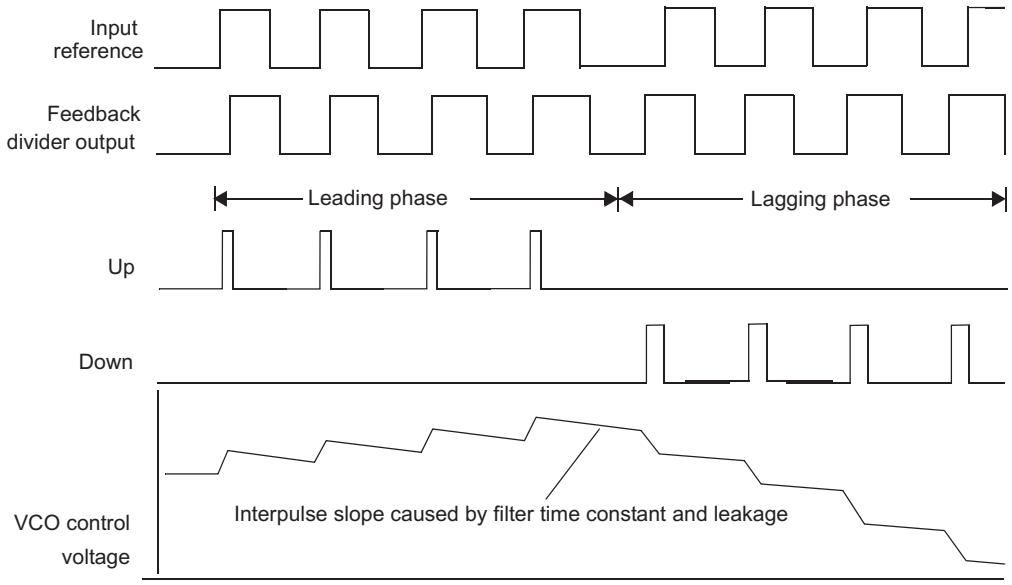
Figure 9-9. Basic PLL Circuit



9.7.1 Phase-Frequency Detector

The phase-frequency detector (PFD) compares the input reference phase/frequency to the phase/frequency of the feedback divider and generates two signals: an *up* pulse and a *down* pulse that drive a charge pump. The resulting charge, when integrated by the circuit at the LF pin, provides a VCO control voltage, as shown in Figure 9-10.

Figure 9-10. PFD Timing



The width of the up pulse and the down pulse depends on the difference in phase between the two inputs. For example, when the reference input leads the feedback input by 10 ns, then an up pulse of approximately 10 ns is generated (see Figure 9-10). On the other hand, when the reference input lags the feedback input by 10 ns, then a down pulse of approximately 10 ns is generated. When the two inputs are exactly in phase, the up pulse and down pulse become essentially zero-width. These pulses are fed to the charge pump block, which meters charge into the low-pass loop filter.

The advantage of a phase-frequency detector over a phase-only detector is that it cannot lock to a harmonic or subharmonic of the reference. This important property also ensures that the output frequency of the VCO is always exactly 2*NF times the reference frequency.

The reference feedback frequency is based upon the VCO frequency and the feedback divider. Fractional multiplication is achieved by changing the feedback divider real-time in order to create the fractional multiplication. As an example, if a multiplier of 100.5 is selected, the feedback divider divides by 100 and 101 in equal proportions; in this case, the PLLMUL bit field would be programmed as 99.5 (0x6380). This fractional multiplication is useful when trying to achieve final frequencies that are non-integer to the input frequency (for example, a final frequency that is a prime number). The fractional portion of the divider should be small compared to the multiplier and so it is recommended that the fractional portion relate to parts in 16, implying that the last 4 bits should always be 0.

9.7.2 Charge Pump and Loop Filter

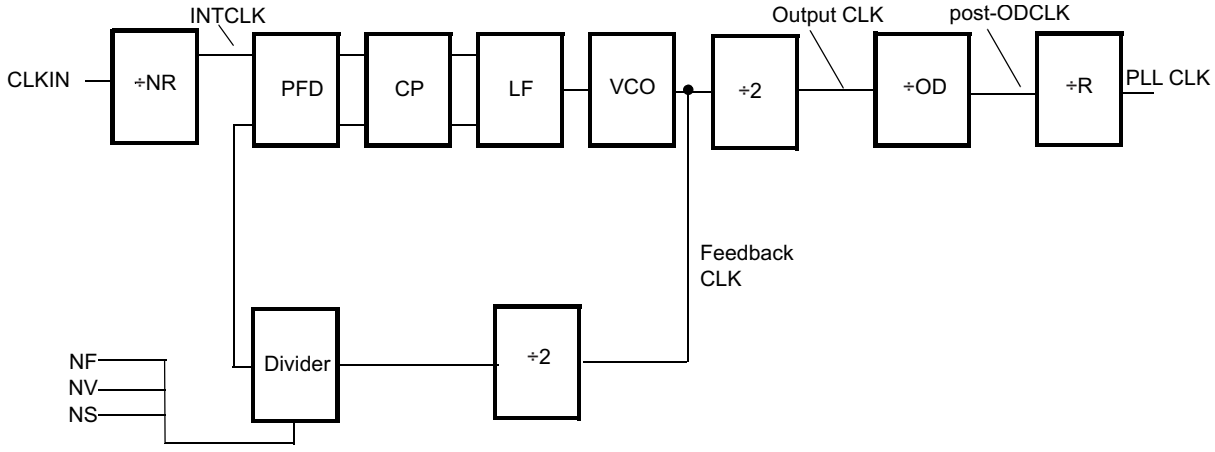
The charge pump (CP) add or remove charge from the loop filter based on the pulses coming from the phase-frequency detector (PFD).

Two components of the filter output signal are summed together: an integral component and a proportional component. The integral component maintains a DC level going to the VCO to set its frequency, and the proportional component makes the VCO track changes in phase to minimize jitter. The capacitors and resistors required for the filter are integrated in silicon.

9.7.3 Voltage-Controlled Oscillator

The output frequency of the VCO is proportional to its input control voltage, which is generated by the charge pump via the integrated loop filter. If the VCO oscillates too slowly, the feedback phase begins to lag the reference phase at the PFD, which increases the control voltage at the VCO. Conversely, if the VCO oscillates too fast, the feedback phase begins to lead the reference phase at the PFD, which decreases the control voltage at the VCO. These two actions keep the VCO running at the correct frequency multiple of the reference.

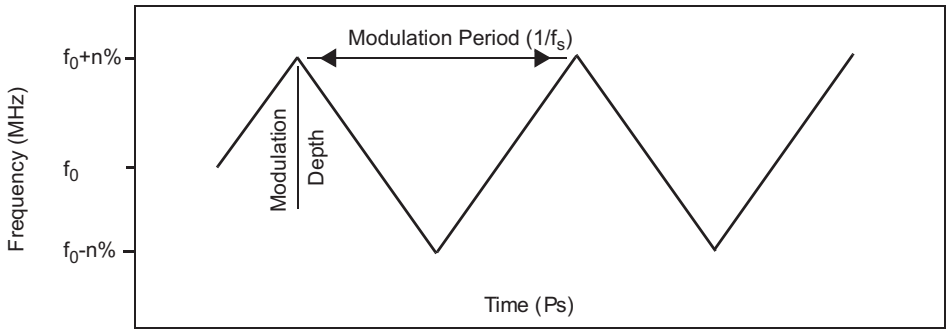
Figure 9-11. PLL Modulation Block Diagram



9.7.4 Frequency Modulation

The output clock of the PLL changes frequency in a controlled way, centered around the unmodulated output frequency. The modulation block directly modulates the VCO frequency at the loop filter, and creates the triangular frequency modulation (see Figure 9-12).

Figure 9-12. Frequency versus Time



9.8 Programming Example

This section provides an example of how to program the PLL. For non-modulation settings, the PLLCTL1 and PLLCTL2 settings from 130 nm process devices can be used without modification.

Suppose that, using a 20 MHz crystal, the application requires:

- 180 MHz GCLK (and HCLK) frequency
- 100 kHz spreading frequency
- 0.5% spreading depth

1. Choose an NR and NS such that:
   - $$\frac{f_{CLKIN}}{NR \times f_s} \geq 40 \quad (14)$$
   - $$f_s \equiv \frac{f_{CLKIN}}{2 \times NR \times NS} \quad (15)$$
   - $$2 \times NS = \frac{f_{CLKIN}}{NR \times f_s} \geq 40 \quad (16)$$
   - (NR,NS) = {(5,20), (4,25), (2,50), (1,100)}
   - Either NR = 5 and NS = 20 or NR = 4 and NS = 25 are reasonable. Another choice (NR = 3 and NS = 33) is possible, if the modulation frequency can vary from 100 KHz.
2. Choose Output CLK frequency as integer divider of output frequency near to 330 MHz. Output CLK frequency shall not exceed 550 MHz or fall below 150 MHz.

   The integer values for 180 MHz are 360 MHz or 540 MHz. 360 MHz is close to the target frequency of 330 MHz and we use this frequency.
3. In this case, either of the following equations are suitable choices for getting to 360 MHz. Choose NR = 5, NS = 20 and set NF = 90 or choose NR = 4, NS = 25 and set NF = 72.

$$\frac{f_{CLKIN}}{NR} = \frac{20[MHz]}{4} = 5[MHz] \ \ or \ \ \frac{f_{CLKIN}}{NR} = \frac{20[MHz]}{5} = 4[MHz] \quad (17)$$

4. Select the output divider OD so that the post-ODCLK frequency does not exceed the maximum frequency of output divider R (device-specific frequency). In this case, choose OD = 2 and R = 1.

5. Compute the divider value NV:

$$Depth = \frac{0.5}{100} = \frac{NV}{NF} \times \frac{NS}{2} = \frac{NV}{90} \times \frac{20}{2} \quad (18)$$

NV = 0.045

6. If it is important to maintain the same average frequency in modulation as in non-modulation, either NF should be modified OR program the MULMOD bit field. The modulation fields create a multiplier offset equal to:

$$\Delta NF = \frac{NV \times NS}{2} \quad (19)$$

If using MULMOD[8:0], then:

$$\Delta NF = \frac{MULMOD[8...0]}{256} = \frac{NV \times NS}{2} = \frac{0.045 \times 20}{2} \quad (20)$$

$$MULMOD[8...0] = \frac{0.045 \times 20}{2} \times 256 = 115.2 \quad (21)$$

MULMOD will be set to 115.

7. Convert the PLL parameters into bit field values:
   - NR = 5, implies that REFCLKDIV[5:0] = 4
   - NS = 20, implies that SPRATE[8:0] = 19 = 0x13
   - NF = 90, implies that PLLMUL[15:0] = 0x5900
   - OD = 2, implies that ODPLL[2:0] = 1
   - R = 1, implies that PLLDIV[4:0] = 0
   - NV = 0.045, implies that SPR_AMOUNT = 91 = 0x5B
   - MULMOD[8:0] = 115 = 0x73
8. Setting only these fields (that is, not BPOS, ROF, or ROS) yields:

   PLLCTL1 = 0x00045900

   PLLCTL2 = 0x04C7325B

   When FMENA is turned on, PLLCTL2 = 0x84C7325B

   The Output CLK is centered in the range from 150 MHz to 550 MHz at 360 MHz.

   NF = 90 falls within the multiplier range from 1 to 256.

   OD is selected so that post-ODCLK meets the device specification.

Dual-Clock Comparator (DCC) Module

This chapter describes the dual-clock comparator (DCC) module.

Topic **Page**

10.1 Introduction

The microcontroller includes one dual-clock comparator (DCC) module. The primary purpose of a DCC module is to measure the frequency of a clock signal using a second clock signal as a reference. An application can use this capability to further enhance the system safety.

10.1.1 Main Features

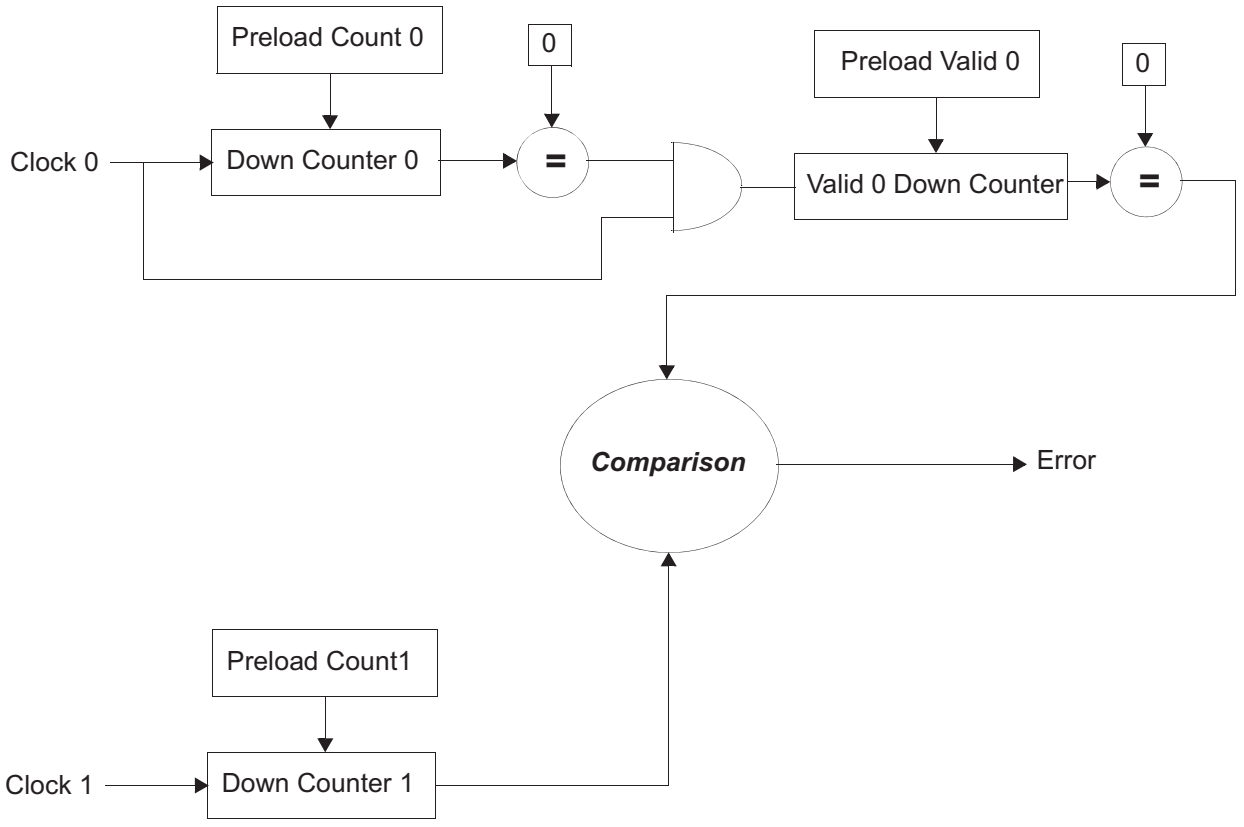
The main features of the DCC module are:

- Allows application to ensure that a fixed ratio is maintained between frequencies of two clock signals
- Supports the definition of a programmable tolerance window in terms of number of reference clock cycles
- Supports continuous monitoring without requiring application intervention
- Alternatively can be used in a single-sequence mode for spot measurements
- Flexible clock source selection for counter0 and counter1 resulting in several specific use cases

10.1.2 Block Diagram

Figure 10-1 illustrates the main concept of the DCC module.

Figure 10-1. DCC Block Diagram



10.2 Module Operation

As shown in Figure 10-1, the DCC contains two counters – counter0 and counter1, which are driven by two signals – clock0 and clock1. The application programs the seed values for both these counters. The application also configures the tolerance window time by configuring the valid counter for clock0.

Counter0 and counter1 both start counting simultaneously when the DCC is enabled. When counter0 counts down to zero, it triggers the count down of the tolerance window counter called valid0.

The seed values of counter0 and counter1 are selected such that if the actual frequencies of clock0 and clock1 are equal to their expected frequencies, then the counter1 will reach zero either at the same time as counter0 or during the count down of the tolerance window counter.

10.2.1 Error Conditions

An error condition is generated by any one of the following:

1. Counter1 counts down to 0 before Counter0 reaches 0.
2. Counter1 does not reach 0 even when Counter0 and Valid0 have both reached 0.
3. Clock0 is not present.
4. Clock1 is not present.

Any error causes the counters to stop counting. An application may then read out the counter values to help determine what caused the error.

The counters can also get reloaded with the seed values and continue counting down under the following conditions:

- The module is reset or restarted by the application, OR
- Counter0, Valid 0 and Counter1 all reach 0 without any error

Figure 10-2. Counter Relationship

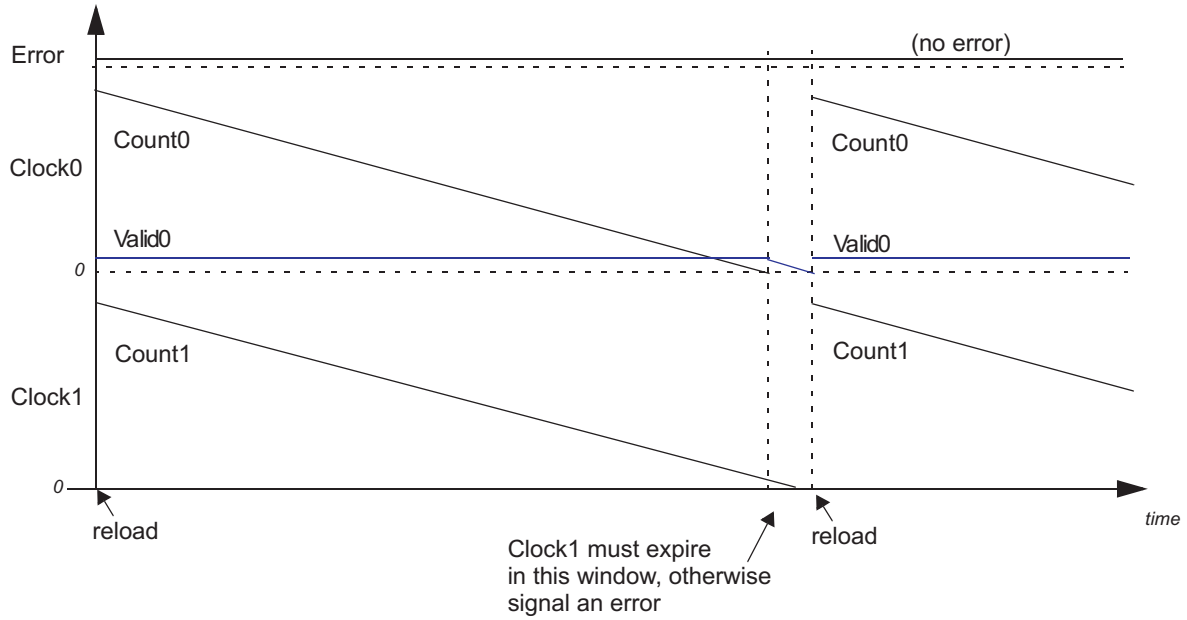


Figure 10-3. Clock1 Slower Than Clock0 - Results in an Error and Stops Counting

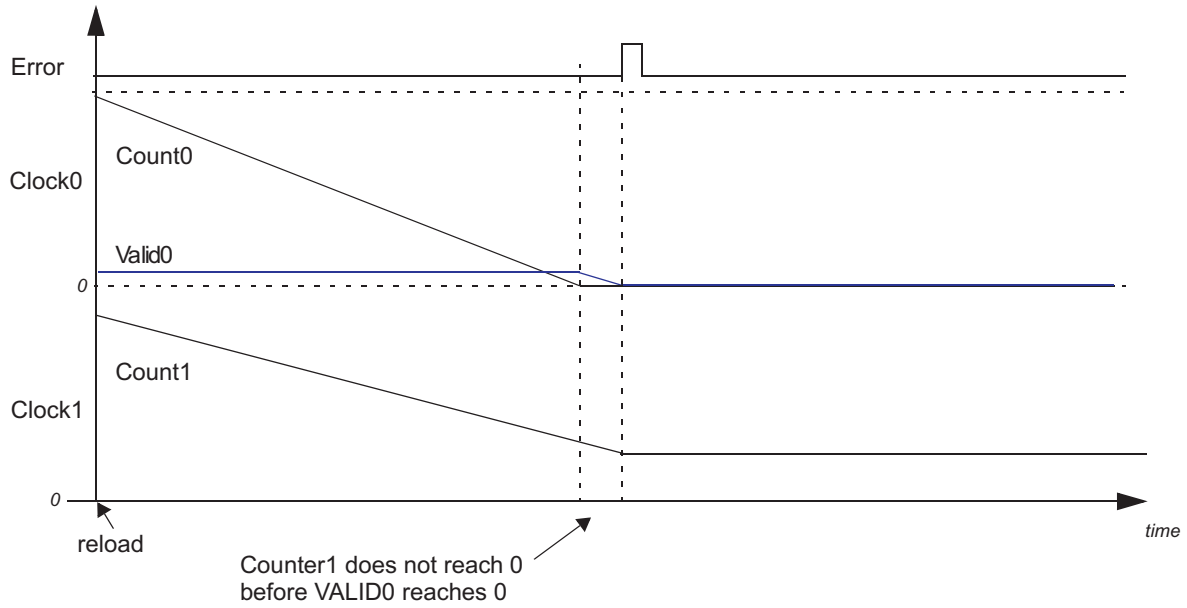


Figure 10-4. Clock1 Faster Than Clock0 - Results in an Error and Stops Counting

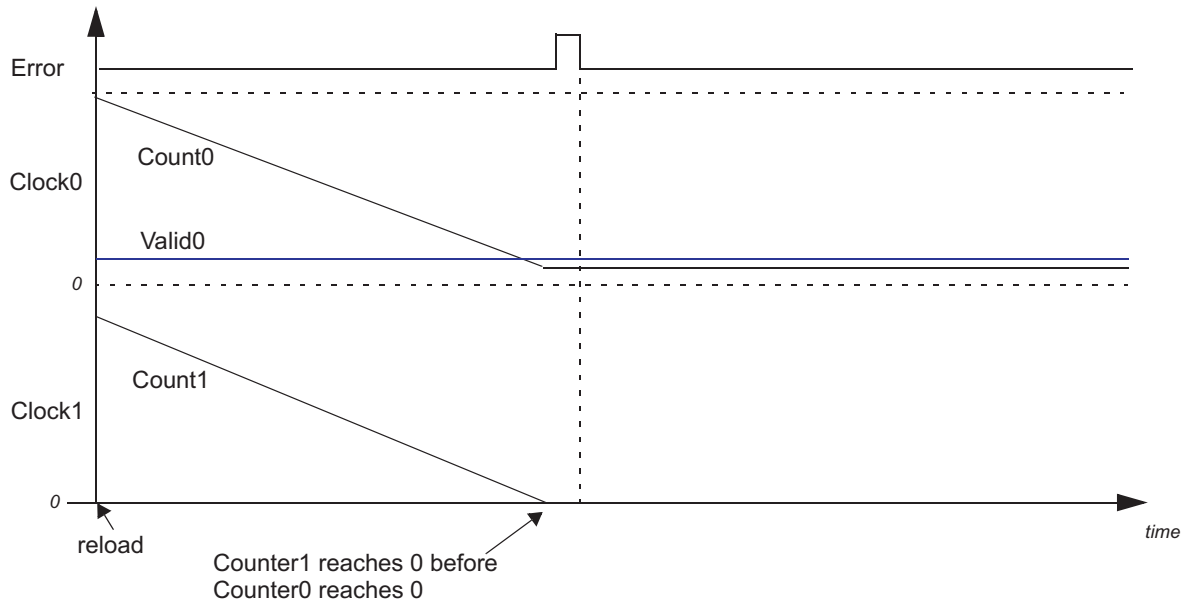


Figure 10-5. Clock1 Not Present - Results in an Error and Stops Counting

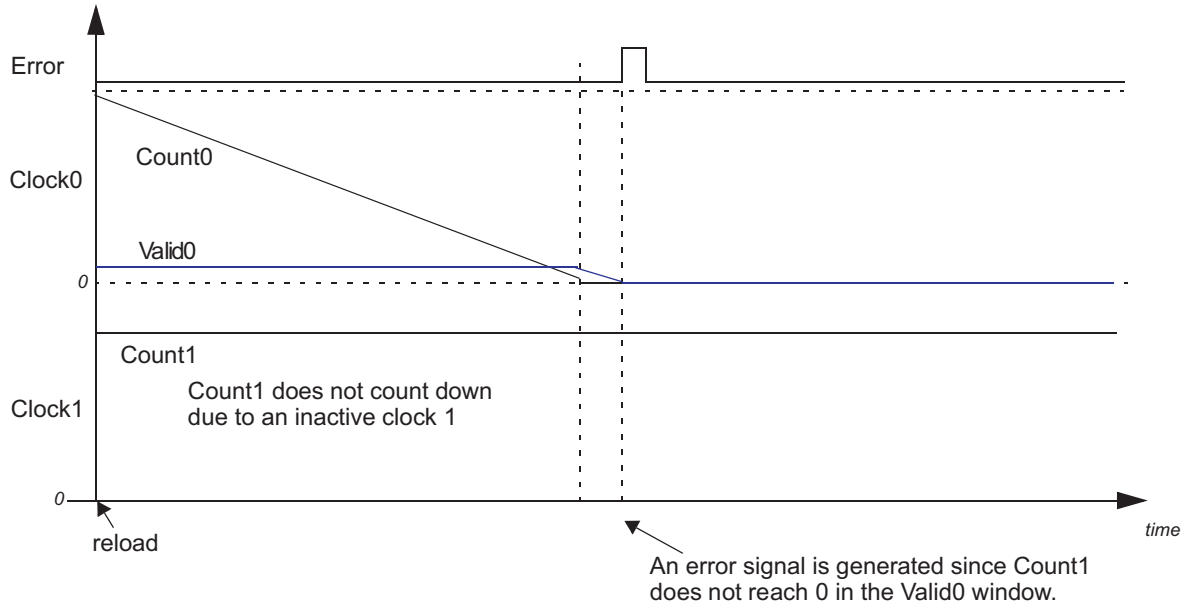
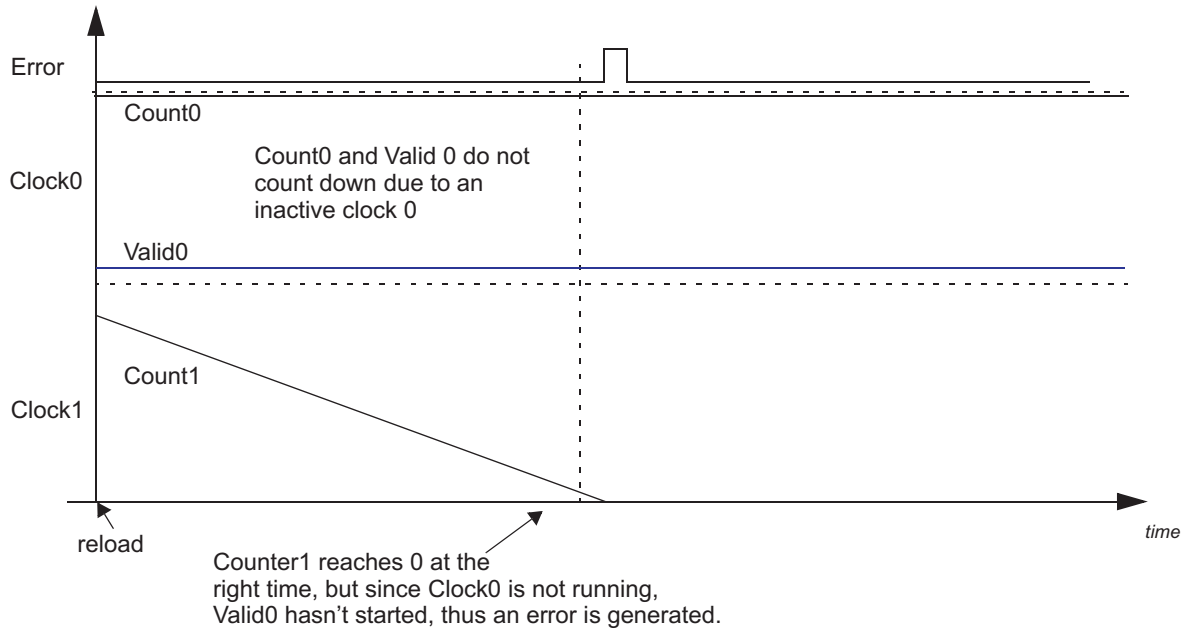


Figure 10-6. Clock0 Not Present - Results in an Error and Stops Counting



10.2.2 Single-Shot Operation Mode

The DCC module can be programmed to count down one time by enabling the single-shot mode. In this mode, the DCC stops operating when the down counter0 and the valid counter0 reach 0.

At the end of one sequence of counting down in this single-shot mode, the DCC gets disabled automatically, which prevents further counting.

If there is no error generated at the end of the sequence, then the DONE status flag is set and a DONE interrupt is generated. The application must clear the DONE flag before restarting the DCC.

If an error is generated at the end of the sequence, then the ERROR status flag is set. The application must clear the ERROR status flag before restarting the DCC.

10.3 Clock Source Selection for Counter0 and Counter1

The DCC module has two counters. Table 10-1 and Table 10-2 show the options for selecting the clock sources for both counters on the DCC module.

CNT0_CLKSRC is a control field in the DCCCNT0CLKSRC register. KEY and CNT1_CLKSRC are control fields in the DCCCNT1CLKSRC register.

Table 10-1. Clock Source Selection for DCC Counter0

CNT0 CLKSRC	Clock Name	Clock Description
5h	HF LPO	High-Frequency Output of Internal Low-Power Oscillator
Ah	TCK	Test Clock
All other values	OSCIN	Main oscillator input

Table 10-2. Clock Source Selection for DCC Counter1

KEY	CNT1 CLKSRC	Clock Name	Clock Description
Ah	0	PLL1 Output	Output from PLL#1
	1h	Reserved	Reserved
	2h	LF LPO	Low-Frequency output of internal Low-Power Oscillator
	3h	HF LPO	High-Frequency output of internal Low-Power Oscillator
	4h	Reserved	Reserved
	5h	EXTCLKIN	External clock input source
	6h	Reserved	Reserved
	7h	Reserved	Reserved
	8h–Fh	VCLK	Peripheral bus clock
All other values	Don't care	N2HET[31]	Signal on the N2HET[31] channel of the N2HET module, either by N2HET program execution, or by the application toggling the N2HET[31] pin as general-purpose output.

NOTE: N2HET[31] can be configured to be brought to the pin using the IOMM module or remain an internal signal. If this signal is assigned to a pin, the application may configure this pin as an output to generate a clock for external use and monitor this clock using the N2HET[31] since the N2HET[31] input buffer is routed to the DCC. Likewise, N2HET[31] can be selected as the DCC clock source utilizing the internal N2HET[31] channel and still be configured as a physical pin with the IOMM and used as an externally available GPIO. However, N2HET[31] cannot be configured as an external input to monitor an external signal such as a clock source somewhere else in the system. For this last case, it is recommended to utilize the External Clock input pin (selectable in place of GIOA[5] by the IOMM module).

10.4 DCC Control Registers

Table 10-3 lists the dual-clock comparator (DCC) module control and status registers. The registers support 8-bit, 16-bit, or 32-bit writes and are aligned on a word (32-bit) boundary. Table 10-3 shows address offsets from the module base address. The base address for DCC registers is FFFF EC00h.

Table 10-3. DCC Registers

Offset	Acronym	Register Description	Section
00h	DCCGCTRL	DCC Global Control Register	Section 10.4.1
04h	DCCREV	DCC Revision ID Register	Section 10.4.2
08h	DCCCNT0SEED	DCC Counter0 Seed Register	Section 10.4.3
0Ch	DCCVALID0SEED	DCC Valid0 Seed Register	Section 10.4.4
10h	DCCCNT1SEED	DCC Counter1 Seed Register	Section 10.4.5
14h	DCCSTAT	DCC Status Register	Section 10.4.6
18h	DCCDCNT0	DCC Counter0 Value Register	Section 10.4.7
1Ch	DCCVALID0	DCC Valid0 Value Register	Section 10.4.8
20h	DCCDCNT1	DCC Counter1 Value Register	Section 10.4.9
24h	DCCCNT1CLKSRC	DCC Counter1 Clock Source Selection Register	Section 10.4.10
28h	DCCCNT0CLKSRC	DCC Counter0 Clock Source Selection Register	Section 10.4.11

10.4.1 DCC Global Control Register (DCCGCTRL)

Figure 10-7 and Table 10-4 describe the DCC Global Control register.

Figure 10-7. DCC Global Control Register (DCCGCTRL) (offset = 00h)

31	16
Reserved	
R-0	

15 12	11 8	7 4	3 0
DONE_INT_ENA	SINGLE_SHOT	ERR_ENA	DCC_ENA
R/WP-5h	R/WP-5h	R/WP-5h	R/WP-5h

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 10-4. DCC Global Control Register (DCCGCTRL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-12	DONE_INT_ENA		Done Interrupt Enable.
			Any operation mode read, privileged mode write:
		5h	No interrupt is generated when the DONE flag is set in the DCC Status (DCCSTAT) register.
		All other values	DONE interrupt is generated when the DONE flag is set in the DCC Status (DCCSTAT) register.
11-8	SINGLE_SHOT		Single-Shot Mode Enable.
			Any operation mode read, privileged mode write:
		Ah	DCC stops counting when counter0 and valid0 both reach 0.
		Bh	DCC stops counting when counter1 reaches 0.
		All other values	DCC counts continuously and only stops when an error occurs.
7-4	ERR_ENA		Error Interrupt Enable.
			Any operation mode read, privileged mode write:
		5h	No interrupt is generated when the ERR flag is set in the DCC Status (DCCSTAT) register.
		All other values	ERROR interrupt is generated when the ERR flag is set in the DCC Status (DCCSTAT) register.
3-0	DCC_ENA		DCC Enable.
			Any operation mode read, privileged mode write:
		5h	All DCC counters are stopped and error-checking is disabled. When an error occurs, the counters stop and this field is set to 5h automatically disabling the DCC counter in hardware.
		All other values	Read: Counters are enabled.
			Write: Load counters with their seed values and begin counting. It is recommended to write Ah to enable counters to protect against single-bit errors.

10.4.2 DCC Revision ID Register (DCCREV)

Figure 10-8 and Table 10-5 describe the DCC Revision ID register.

Figure 10-8. DCC Revision ID Register (DCCREV) (offset = 04h)

31–30	29–28	27–16
SCHEME	Reserved	FUNC
R-1	R-0	R-0

15–11	10–8	7–6	5–0
RTL	MAJOR	CUSTOM	MINOR
R-0	R-2h	R-0	R-4h

LEGEND: R = Read only; -*n* = value after reset

Table 10-5. DCC Revision ID Register (DCCREV) Field Descriptions

Bit	Field	Value	Description
31-30	SCHEME	1	Reads return 1, writes have no effect.
29-28	Reserved	0	Reads return 0. Writes have no effect.
27-16	FUNC	0	Functional release number. Reads return 0, writes have no effect.
15-11	RTL	0	Design release number. Reads return 0, writes have no effect.
10-8	MAJOR	2h	Major revision number. Reads return 2h, writes have no effect.
7-6	CUSTOM	0	Custom version number. Reads return 0, writes have no effect.
5-0	MINOR	4h	Minor revision number. Reads return 4h, writes have no effect.

10.4.3 DCC Counter0 Seed Register (DCCCNT0SEED)

Figure 10-9 and Table 10-6 describe the DCC Counter0 Seed register.

NOTE: Seed for Counter0 must be non-zero

The DCC must only be enabled after programming a non-zero value in the counter0 seed register.

Figure 10-9. DCC Counter0 Seed Register (DCCCNT0SEED) (offset = 08h)

31–20	19–16
Reserved	COUNT0_SEED
R-0	R/WP-0

15–0
COUNT0_SEED
R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 10-6. DCC Counter0 Seed Register (DCCCNT0SEED) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	COUNT0_SEED	0-FFFFFh	Seed value for DCC counter0. Reads in any operating mode return the current value of counter0. Writing in privileged mode only sets the current seed value for counter0.

10.4.4 DCC Valid0 Seed Register (DCCVALID0SEED)

Figure 10-10 and Table 10-7 describe the DCC Valid0 Seed register.

NOTE: Seed for Valid0 must be at least 4h

The DCC must only be enabled after programming a value greater than or equal to 4h in the valid0 seed register.

Figure 10-10. DCC Valid0 Seed Register (DCCVALID0SEED) (offset = 0Ch)

31	16
Reserved	
R-0	

15	0
VALID0_SEED	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 10-7. DCC Valid0 Seed Register (DCCVALID0SEED) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	VALID0_SEED	0-FFFFh	Seed value for DCC Valid0. This value defines the window within which the counter1 must reach 0. This window needs to be at least 4 cycles wide. Reads in any operating mode return the current value of Valid0. Writing in privileged mode only sets the current seed value for Valid0.

10.4.5 DCC Counter1 Seed Register (DCCCNT1SEED)

Figure 10-11 and Table 10-8 describe the DCC Counter1 Seed register.

NOTE: Seed for Counter1 must be non-zero

The DCC must only be enabled after programming a non-zero value in the counter1 seed register.

Figure 10-11. DCC Counter1 Seed Register (DCCCNT1SEED) (offset = 10h)

31	20	19	16
Reserved		COUNT1_SEED	
R-0		R/WP-0	

15	0
COUNT1_SEED	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 10-8. DCC Counter1 Seed Register (DCCCNT1SEED) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	COUNT1_SEED	0-FFFFFh	Seed value for DCC counter1. Reads in any operating mode return the current value of counter1. Writing in privileged mode only sets the current seed value for counter1.

10.4.6 DCC Status Register (DCCSTAT)

Figure 10-7 and Table 10-4 describe the DCC Status register.

Figure 10-12. DCC Status Register (DCCSTAT) (offset = 14h)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		DONE	ERR
R-0		R/W1CP-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 10-9. DCC Status Register (DCCSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	DONE FLG		Single-Shot Sequence Done flag. Indicates that a single-shot DCC sequence is done without any error.
		0	Read: Single-shot sequence is not done. Write: Writing 0 has no effect.
		1	Read: Single-shot sequence is done without any error. Write: Writing 1 in privileged mode clears the DONE flag.
0	ERR FLG		Error flag. Indicates that a DCC error has occurred.
		0	Read: DCC error has not occurred. Write: Writing 0 has no effect.
		1	Read: An error has occurred. Write: Writing 1 in privileged mode clears the ERR flag.

10.4.7 DCC Counter0 Value Register (DCCCNT0)

Figure 10-13 and Table 10-10 describe the DCC Counter0 Value register.

NOTE: Reads may not return exact current value of Counter0

Reading the counter0 value while counting is enabled may not return the exact value of the counter0.

Figure 10-13. DCC Counter0 Value Register (DCCCNT0) (offset = 18h)

31	20	19	16
Reserved		COUNT0	
R-0		R-0	

15	0
COUNT0	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 10-10. DCC Counter0 Value Register (DCCCNT0) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	COUNT0	0-FFFFFh	Current value of DCC counter0. Reads in any operating mode return the current value of counter0. Writes have no effect.

10.4.8 DCC Valid0 Value Register (DCCVALID0)

Figure 10-14 and Table 10-11 describe the DCC Valid0 Value register.

NOTE: Reads may not return exact current value of Valid0

Reading the valid0 value while counting is enabled may not return the exact value of the valid0.

Figure 10-14. DCC Valid0 Value Register (DCCVALID0) (offset = 1Ch)

31	16
Reserved	
R-0	

15	0
VALID0	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 10-11. DCC Valid0 Value Register (DCCVALID0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	VALID0	0-FFFFh	Current value for DCC Valid0. Reads in any operating mode return the current value of Valid0. Writes have no effect.

10.4.9 DCC Counter1 Value Register (DCCCNT1)

Figure 10-15 and Table 10-12 describe the DCC Counter1 Value register.

NOTE: Reads may not return exact current value of Counter1

Reading the counter1 value while counting is enabled may not return the exact value of the counter1.

Figure 10-15. DCC Counter1 Value Register (DCCCNT1) (offset = 20h)

31	20	19	16
Reserved		COUNT1	
R-0		R/WP-0	

15	0
COUNT1	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 10-12. DCC Counter1 Value Register (DCCCNT1) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	COUNT1	0-FFFFFh	Current value for DCC Counter1. Reads in any operating mode return the current value of counter1. Writes have no effect.

10.4.10 DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC)

Figure 10-15 and Table 10-12 describe the DCC Counter1 Value register.

Figure 10-16. DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) (offset = 24h)

31	16
Reserved	
R-0	

15	12	11	4	3	0
KEY		Reserved		CNT1_CLKSRC	
R/WP-5h		R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 10-13. DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-12	KEY		Key to enable clock source selection for counter1. Reads in any operating mode return the current value of the key. Writing in privileged mode sets the key value.
		Ah	Writing Ah as the key enables the CNT1_CLKSRC field to define the clock source for counter1.
		All other values	Writing any other value as the key disables the clock source selection for counter1. In this case, the N2HET signal is used as the source for counter1.
11-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	CNT1_CLKSRC	0-Fh	Clock Source for counter1 when KEY is programmed to be Ah. See Table 10-2. Reads in any operating mode return the current value of CLKSRC. Writes in privileged mode select the clock source for counter1.

10.4.11 DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC)

Figure 10-15 and Table 10-12 describe the DCC Counter0 Value register.

Figure 10-17. DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC) (offset = 28h)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		CNT0_CLKSRC	
R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 10-14. DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	CNT0_CLKSRC	0-Fh	Clock Source for counter0. See Table 10-1. Reads in any operating mode return the current value of CLKSRC. Writes in privileged mode select the clock source for counter0.

Error Signaling Module (ESM)

This chapter provides the details of the error signaling module (ESM) that aggregates device errors and provides the capability to define internal and external error response based on error severity.

Topic **Page**

11.1 Overview

The Error Signaling Module (ESM) collects and reports the various error conditions on the microcontroller. The error condition is categorized based on a severity level. Error response is then generated based on the category of the error. Possible error responses include a low-priority interrupt, high-priority interrupt, and an external pin action.

11.1.1 Features

- Up to 128 error channels are supported, divided into 3 different groups:
  - 64 Group1 (low-severity) channels with configurable interrupt generation and configurable $\overline{\text{ERROR}}$ pin behavior
  - 32 Group2 (high-severity) channels with predefined interrupt generation and predefined $\overline{\text{ERROR}}$ pin behavior
  - 32 Group3 (high-severity) channels with no interrupt generation and predefined $\overline{\text{ERROR}}$ pin behavior. These channels have no interrupt response as they are reserved for CPU based diagnostics that generate aborts directly to the CPU.
- Dedicated device $\overline{\text{ERROR}}$ pin to signal an external observer
- Configurable timebase for $\overline{\text{ERROR}}$ pin output
- Error forcing capability for latent fault testing

11.1.2 Block Diagram

As shown in Figure 11-1, the ESM channels are divided into three groups. Group1 channels are considered to be low-severity. Group1 errors have a configurable interrupt response and configurable $\overline{\text{ERROR}}$ pin behavior. Note that the ESM Status Register 1 (ESMSR1) for error group1 gets updated, regardless of whether an ESM interrupt for that Group1 channel is enabled or not. Group2 channels are connected to higher-severity error signals. Group2 errors generate a non-maskable high-priority interrupt to the CPU and assert the $\overline{\text{ERROR}}$ pin. Group3 channels indicate errors of the highest severity. Check the specific part's datasheet for identifying group3 errors and their expected responses. Group3 errors always generate an $\overline{\text{ERROR}}$ pin output.

The ESM interrupt and $\overline{\text{ERROR}}$ pin behavior are also summarized in Table 11-1.

Figure 11-1. Block Diagram

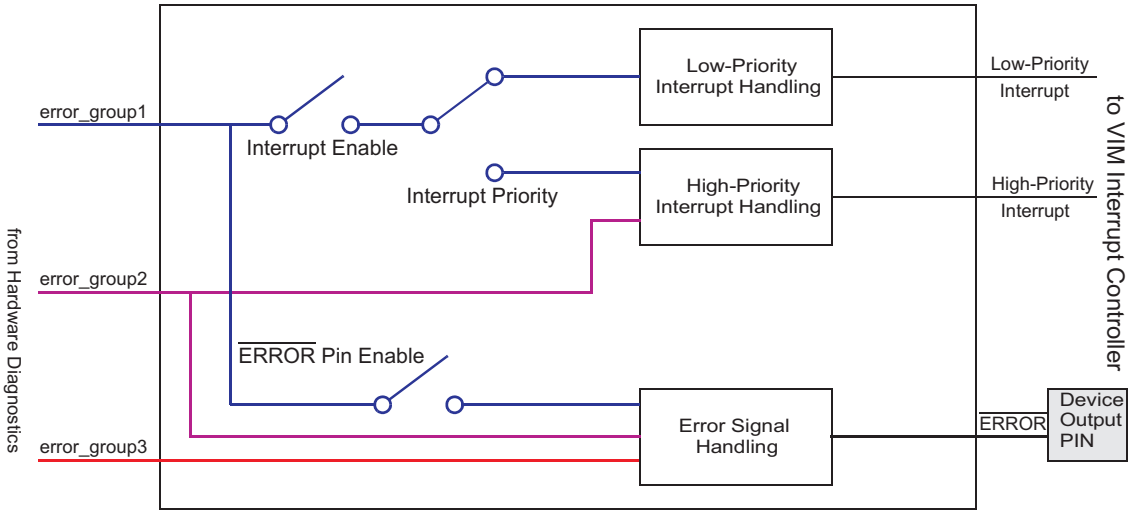


Table 11-1. ESM Interrupt and $\overline{\text{ERROR}}$ Pin Behavior

Error Group	Interrupt to CPU	Interrupt Priority	$\overline{\text{ERROR}}$ Pin Response
1	Can be enabled or disabled for each channel	Can be selected as low/high-priority for each channel	$\overline{\text{ERROR}}$ pin action can be selected for each channel separately
2	Cannot be disabled	High priority	$\overline{\text{ERROR}}$ pin is asserted
3	No interrupt	NA	$\overline{\text{ERROR}}$ pin is asserted

Figure 11-2 and Figure 11-3 show the interrupt response handling and $\overline{\text{ERROR}}$ pin response handling with register configuration. The total active time of the $\overline{\text{ERROR}}$ pin is controlled by the Low-Time Counter Preload register (LTCP) and the key register (ESMEPSR) as shown in Figure 11-3. See Section 11.2.2 for details.

Figure 11-2. Interrupt Response Handling

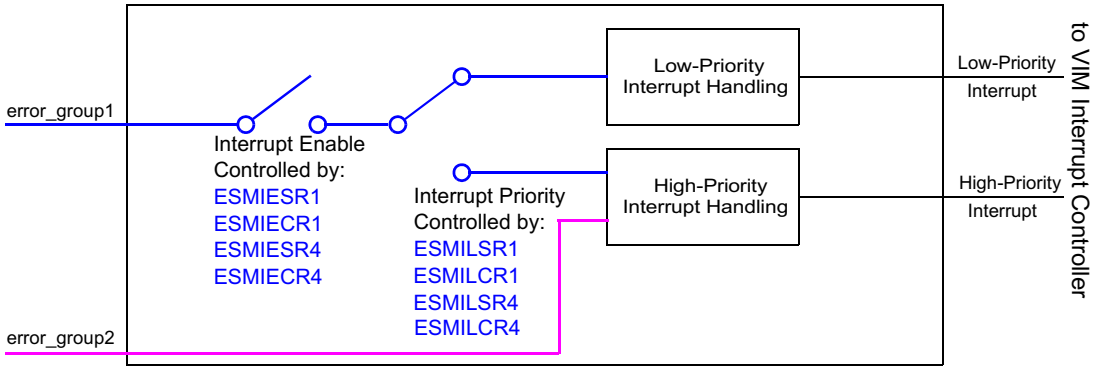
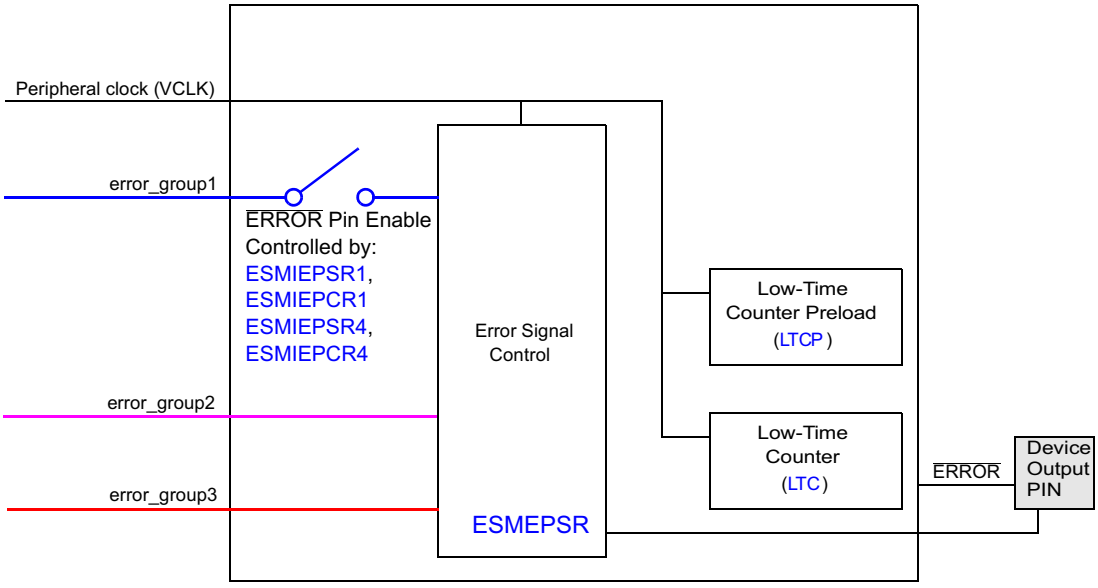


Figure 11-3. $\overline{\text{ERROR}}$ Pin Response Handling



11.2 Module Operation

This device has 128 error channels, divided into 3 different error groups. Please refer to the device datasheet for ESM channel assignment details.

The ESM module has error flags for each error channel. The error status registers ESMSR1, ESMSR4, ESMSR2, ESMSR3 provide status information on a pending error of Group1 (Channel 0-31), Group1 (Channel 32-63), Group2, and Group3, respectively. The ESMEPSR register provides the current $\overline{\text{ERROR}}$ status. The module also provides a status shadow register, ESMSSR2, which maintains the error flags of Group2 until power-on reset ($\overline{\text{PORRST}}$) is asserted. See Section 11.2.1 for details of their behavior during power on reset and warm reset.

Once an error occurs, the ESM module will set the corresponding error flags. In addition, it can trigger an interrupt, $\overline{\text{ERROR}}$ pin outputs low depending on the ESM settings. Once the $\overline{\text{ERROR}}$ pin outputs low, a power on reset or a write of 5h to ESMEKR is required to release the ESM error pin back to normal state. See Section 11.2.2 for details. The application can read the error status registers (ESMSR1, ESMSR4, ESMSR2, and ESMSR3) to debug the error. If an $\overline{\text{RST}}$ is triggered or the error interrupt has been served, the error flag of Group2 should be read from ESMSSR2 because the error flag in ESMSR2 will be cleared by $\overline{\text{RST}}$.

The user can also test the functionality of the $\overline{\text{ERROR}}$ pin by forcing an error. See Section 11.2.3 for details.

11.2.1 Reset Behavior

Power on reset:

- $\overline{\text{ERROR}}$ pin behavior

  When $\overline{\text{PORRST}}$ is active, the $\overline{\text{ERROR}}$ pin is in a high impedance state (output drivers disabled).
- Register behavior

  After $\overline{\text{PORRST}}$, all registers in ESM module will be re-initialized to the default value. All the error status registers are cleared to zero.

Warm reset ($\overline{\text{RST}}$):

- $\overline{\text{ERROR}}$ pin behavior

  During $\overline{\text{RST}}$, the $\overline{\text{ERROR}}$ pin is in “output active” state with pull-down disabled. The $\overline{\text{ERROR}}$ pin remains unchanged after $\overline{\text{RST}}$.
- Register behavior

  After $\overline{\text{RST}}$, ESMSR1, ESMSR4, ESMSSR2, ESMSR3 and ESMEPSR register values remains unchanged. Since $\overline{\text{RST}}$ does not clear the critical failure registers, the user can read those registers to debug the failures after $\overline{\text{RST}}$ pin goes back to high.

  After $\overline{\text{RST}}$, if one of the flags in ESMSR1 and ESMSR4 is set, the interrupt service routine will be called once the corresponding interrupt is enabled.

NOTE: ESMSR2 is cleared after $\overline{\text{RST}}$. The flag in ESMSR2 gets cleared when reading the appropriate vector in the ESMIOFFHR offset register. Reading ESMIOFFHR will not clear the ESMSR1, ESMSR4, and the shadow register ESMSSR2. Reading ESMIOFFLR will also not clear the ESMSR1 and ESMSR4.

11.2.2 $\overline{\text{ERROR}}$ Pin Timing

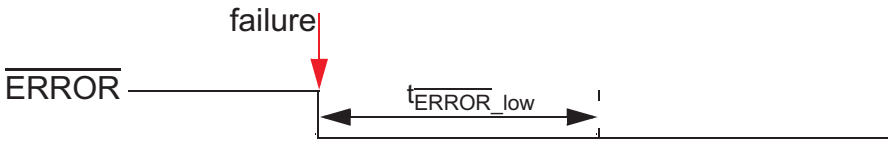
The $\overline{\text{ERROR}}$ pin is an active-low function. The state of the pin is also readable from $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR). The pin is in a high-impedance state during power-on reset. Once the ESM module drives the $\overline{\text{ERROR}}$ pin low, it remains in this state for the time specified by the Low-Time Counter Preload register (LTCPR). Based on the time period of the peripheral clock (VCLK), the total active time of the $\overline{\text{ERROR}}$ pin can be calculated as:

$$t_{\overline{ERROR}\_low} = t_{VCLK} \times (LTCP + 1) \quad (22)$$

Once this period expires, the $\overline{\text{ERROR}}$ pin is set to high in case the reset of the $\overline{\text{ERROR}}$ pin was requested. This request is done by writing an appropriate key (5h) to the key register (ESMEKR) during the $\overline{\text{ERROR}}$ pin low time. Here are a few examples:

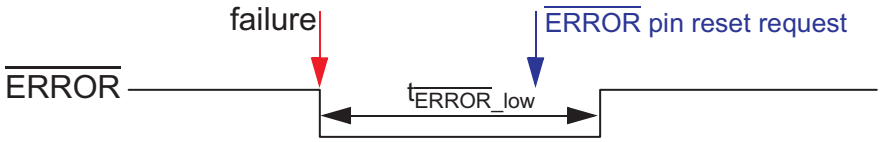
Example 1: ESM detects a failure and drives the $\overline{\text{ERROR}}$ pin low. No $\overline{\text{ERROR}}$ pin reset is requested. The $\overline{\text{ERROR}}$ pin continues outputting low until power on reset occurs.

Figure 11-4. $\overline{\text{ERROR}}$ Pin Timing - Example 1



Example 2: ESM detects a failure and drives the $\overline{\text{ERROR}}$ pin low. An $\overline{\text{ERROR}}$ pin reset request is received before $t_{\overline{ERROR}\_low}$ expires. In this case, the $\overline{\text{ERROR}}$ pin is set to high immediately after $t_{\overline{ERROR}\_low}$ expires.

Figure 11-5. $\overline{\text{ERROR}}$ Pin Timing - Example 2

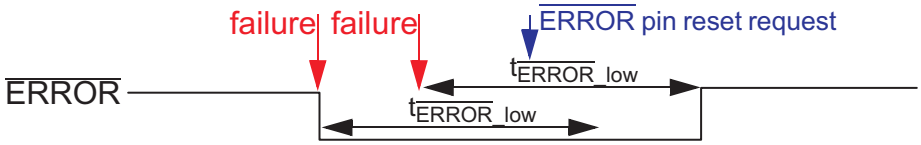


Example 3: ESM detects a failure and drives the $\overline{\text{ERROR}}$ pin low. An $\overline{\text{ERROR}}$ pin reset request is received after $t_{\overline{ERROR}\_low}$ expires. In this case, the $\overline{\text{ERROR}}$ pin is set to high immediately after $\overline{\text{ERROR}}$ pin reset request is received.

Figure 11-6. $\overline{\text{ERROR}}$ Pin Timing - Example 3

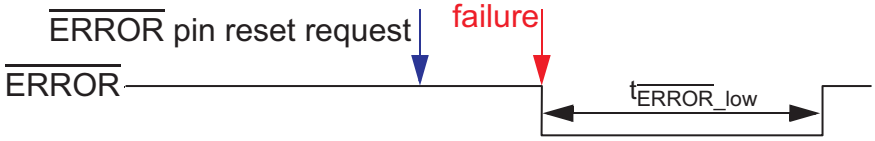
Example 4: ESM detects a failure and drives the $\overline{\text{ERROR}}$ pin low. Another failure occurs within the time the pin stays low. In this case, the low-time counter will be reset when the other failure occurs. In other words, $t_{\overline{\text{ERROR}}\_low}$ should be counted from whenever the most recent failure occurs.

Figure 11-7. $\overline{\text{ERROR}}$ Pin Timing - Example 4



Example 5: The reset of the $\overline{\text{ERROR}}$ pin was requested by the software even before the failure occurs. In this case, the $\overline{\text{ERROR}}$ pin is set to high immediately after $t_{\overline{\text{ERROR}}\_low}$ expires. This case is not recommended and should be avoided by the application.

Figure 11-8. $\overline{\text{ERROR}}$ Pin Timing - Example 5



Example 6: Failure1, then $\overline{\text{ERROR}}$ pin reset request, then Failure2 occurs before $\overline{\text{ERROR}}$ pin gets reset. In this case, the $\overline{\text{ERROR}}$ pin low-time is just extended (restarted) when the failure2 occurs and goes high when this count-down expires. There now is a scenario where the $\overline{\text{ERROR}}$ pin is high and the group2/3 status flag is set. To avoid this scenario, the application must write 5h followed by 0 to the ESM Error Key Register (ESMEKR). In this case, the $\overline{\text{ERROR}}$ pin will go high and then go low again to indicate the second failure.

11.2.3 Forcing an Error Condition

The error response generation mechanism is testable by software by forcing an error condition. This allows testing the $\overline{\text{ERROR}}$ pin functionality. By writing a dedicated key to the ESM Error Key Register (ESMEKR), the $\overline{\text{ERROR}}$ pin is set to low for the specified time. The following steps describe how to force an error condition:

1. Check $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR). This register must be 1 to switch into the error forcing mode.

   The ESM module cannot be switched into the error forcing mode if a failure has already been detected in functional mode. The application command to switch to error forcing mode is ignored.

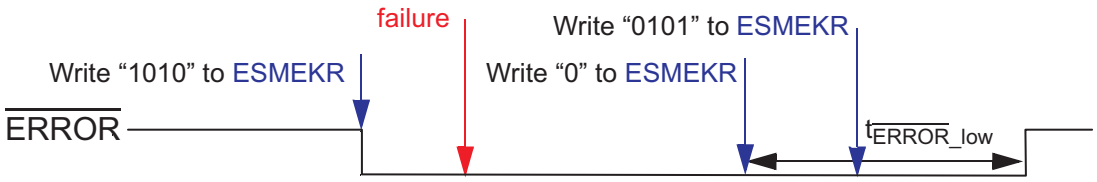
2. Write 5h to the ESM Error Key Register (ESMEKR). After that, the $\overline{\text{ERROR}}$ pin should output low (error force mode).

   Once the application puts the ESM module in the error forcing mode, the $\overline{\text{ERROR}}$ pin cannot indicate the normal error functionality. If a failure occurs during this time, it gets still latched and the LTC is reset and stopped. The error output pin is already driven low on account of the error forcing mode. When the ESM is forced back to normal functional mode, the LTC becomes active and forces the $\overline{\text{ERROR}}$ pin low until the expiration of the LTC (see Figure 11-9).

3. Write 0 to the ESM Error Key Register (ESMEKR) back to the active normal mode.

   If there are no errors detected while the ESM module is in the error forcing mode, the $\overline{\text{ERROR}}$ pin goes high immediately after exiting the error forcing mode.

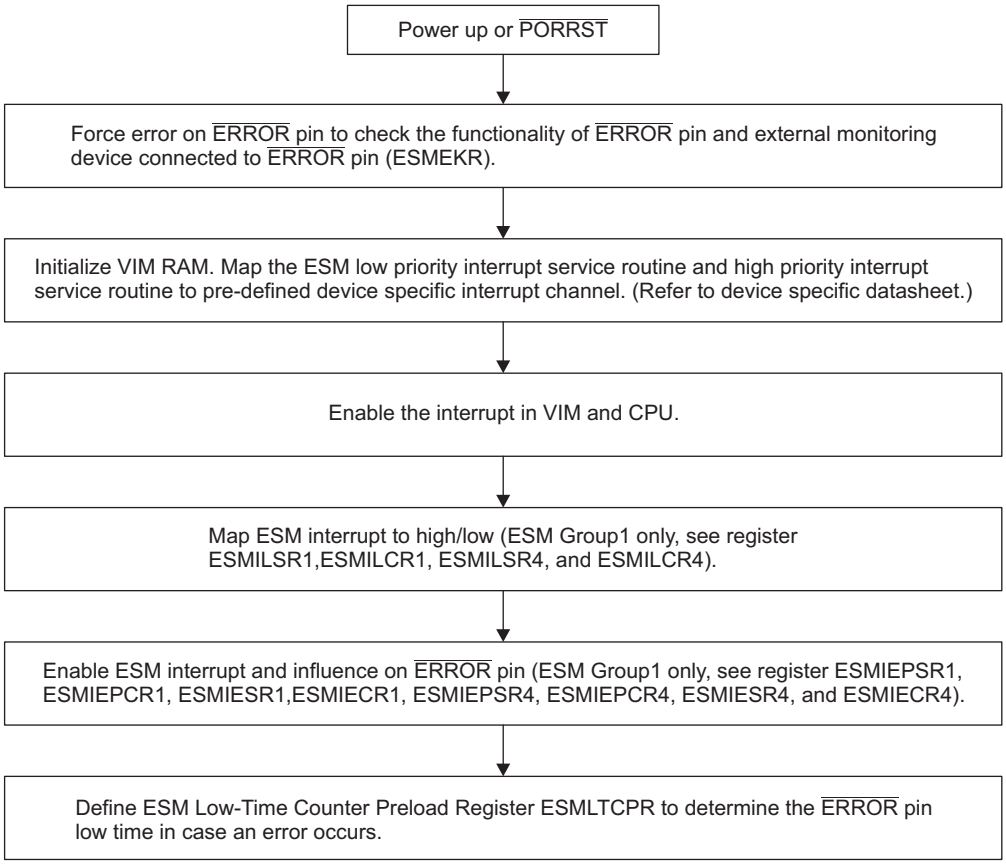
Figure 11-9. $\overline{\text{ERROR}}$ Pin Timing - Example 7



11.3 Recommended Programming Procedure

During the initialization stage, the application code should follow the recommendations in Figure 11-10 to initialize the ESM.

Figure 11-10. ESM Initialization



Once an error occurs, it can trigger an interrupt, $\overline{\text{ERROR}}$ pin outputs low depending on the ESM settings. Once the $\overline{\text{ERROR}}$ pin outputs low, a power on reset or a write of 5h to ESMEKR is required to release the ESM back to normal state. The application can read the error status registers (ESMSR1, ESMSR4, ESMSR2, and ESMSR3) to debug the error. If an $\overline{\text{RST}}$ is triggered or the error interrupt has been served, the error flag of Group2 should be read from ESMSSR2 because the error flag in ESMSR2 will be cleared by $\overline{\text{RST}}$.

11.4 Control Registers

This section describes the ESM registers. Each register begins on a 32-bit word boundary. The registers support 32-bit, 16-bit, and 8-bit accesses. The base address for the registers is FFFF F500h.

Table 11-2. ESM Module Registers

Address	Acronym	Register Description	Section
FFFF F500h	ESMEEPAPR1	ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1	Section 11.4.1
FFFF F504h	ESMDEPAPR1	ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1	Section 11.4.2
FFFF F508h	ESMIESR1	ESM Interrupt Enable Set Register 1	Section 11.4.3
FFFF F50Ch	ESMIECR1	ESM Interrupt Enable Clear Register 1	Section 11.4.4
FFFF F510h	ESMILSR1	Interrupt Level Set Register 1	Section 11.4.5
FFFF F514h	ESMILCR1	Interrupt Level Clear Register 1	Section 11.4.6
FFFF F518h	ESMSR1	ESM Status Register 1	Section 11.4.7
FFFF F51Ch	ESMSR2	ESM Status Register 2	Section 11.4.8
FFFF F520h	ESMSR3	ESM Status Register 3	Section 11.4.9
FFFF F524h	ESMEPSR	ESM $\overline{\text{ERROR}}$ Pin Status Register	Section 11.4.10
FFFF F528h	ESMIOFFHR	ESM Interrupt Offset High Register	Section 11.4.11
FFFF F52Ch	ESMIOFFLR	ESM Interrupt Offset Low Register	Section 11.4.12
FFFF F530h	ESMLTCR	ESM Low-Time Counter Register	Section 11.4.13
FFFF F534h	ESMLTCPR	ESM Low-Time Counter Preload Register	Section 11.4.14
FFFF F538h	ESMEKR	ESM Error Key Register	Section 11.4.15
FFFF F53Ch	ESMSSR2	ESM Status Shadow Register 2	Section 11.4.16
FFFF F540h	ESMIEPSR4	ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4	Section 11.4.17
FFFF F544h	ESMIEPCR4	ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4	Section 11.4.18
FFFF F548h	ESMIESR4	ESM Interrupt Enable Set Register 4	Section 11.4.19
FFFF F54Ch	ESMIECR4	ESM Interrupt Enable Clear Register 4	Section 11.4.20
FFFF F550h	ESMILSR4	Interrupt Level Set Register 4	Section 11.4.21
FFFF F554h	ESMILCR4	Interrupt Level Clear Register 4	Section 11.4.22
FFFF F558h	ESMSR4	ESM Status Register 4	Section 11.4.23

11.4.1 ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1)

This register is dedicated for Group1.

Figure 11-11. ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1) [address = FFFF F500h]

31	16
IEPSET	
R/WP-0	
15	0
IEPSET	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 11-3. ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1) Field Descriptions

Bit	Field	Value	Description
31-0	IEPSET		Enable $\overline{\text{ERROR}}$ Pin Action/Response on Group 1. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. Write: Leaves the bit and the corresponding clear bit in the ESMIEPCR1 register unchanged.
		1	Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. Write: Enables failure influence on $\overline{\text{ERROR}}$ pin and sets the corresponding clear bit in the ESMIEPCR1 register.

11.4.2 ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1)

This register is dedicated for Group1.

Figure 11-12. ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1) [address = FFFF F504h]

31	16
IEPCLR	
R/WP-0	
15	0
IEPCLR	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 11-4. ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1) Field Descriptions

Bit	Field	Value	Description
31-0	IEPCLR		Disable $\overline{\text{ERROR}}$ Pin Action/Response on Group 1. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. Write: Leaves the bit and the corresponding set bit in the ESMIEPSR1 register unchanged.
		1	Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. Write: Disables failure influence on $\overline{\text{ERROR}}$ pin and clears the corresponding set bit in the ESMIEPSR1 register.

11.4.3 ESM Interrupt Enable Set Register 1 (ESMIESR1)

This register is dedicated for Group1.

Figure 11-13. ESM Interrupt Enable Set Register 1 (ESMIESR1) [address = FFFF F508h]

31	16
INTENSET	
R/WP-0	
15	0
INTENSET	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 11-5. ESM Interrupt Enable Set Register 1 (ESMIESR1) Field Descriptions

Bit	Field	Value	Description
31-0	INTENSET		Set interrupt enable.
			Read in User and Privileged mode. Write in Privileged mode only.
		0	Read: Interrupt is disabled.
			Write: Leaves the bit and the corresponding clear bit in the ESMIECR1 register unchanged.
		1	Read: Interrupt is enabled.
			Write: Enables interrupt and sets the corresponding clear bit in the ESMIECR1 register.

11.4.4 ESM Interrupt Enable Clear Register 1 (ESMIECR1)

This register is dedicated for Group1.

Figure 11-14. ESM Interrupt Enable Clear Register 1 (ESMIECR1) [address = FFFF F50Ch]

31	16
INTENCLR	
R/WP-0	
15	0
INTENCLR	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 11-6. ESM Interrupt Enable Clear Register 1 (ESMIECR1) Field Descriptions

Bit	Field	Value	Description
31-0	INTENCLR		Clear interrupt enable.
			Read in User and Privileged mode. Write in Privileged mode only.
		0	Read: Interrupt is disabled.
			Write: Leaves the bit and the corresponding set bit in the ESMIESR1 register unchanged.
		1	Read: Interrupt is enabled.
			Write: Disables interrupt and clears the corresponding set bit in the ESMIESR1 register.

11.4.5 ESM Interrupt Level Set Register 1 (ESMILSR1)

This register is dedicated for Group1.

Figure 11-15. ESM Interrupt Level Set Register 1 (ESMILSR1) [address = FFFF F510h]

31	16
INTLVLSET	
R/WP-0	

15	0
INTLVLSET	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 11-7. ESM Interrupt Level Set Register 1 (ESMILSR1) Field Descriptions

Bit	Field	Value	Description
31-0	INTLVLSET		Set interrupt priority. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Interrupt of channel x is mapped to low-level interrupt line. Write: Leaves the bit and the corresponding clear bit in the ESMILCR1 register unchanged.
		1	Read: Interrupt of channel x is mapped to high-level interrupt line. Write: Maps interrupt of channel x to high-level interrupt line and sets the corresponding clear bit in the ESMILCR1 register.

11.4.6 ESM Interrupt Level Clear Register 1 (ESMILCR1)

This register is dedicated for Group1.

Figure 11-16. ESM Interrupt Level Clear Register 1 (ESMILCR1) [address = FFFF F514h]

31	16
INTLVLCLR	
R/WP-0	

15	0
INTLVLCLR	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 11-8. ESM Interrupt Level Clear Register 1 (ESMILCR1) Field Descriptions

Bit	Field	Value	Description
31-0	INTLVLCLR		Clear interrupt priority. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Interrupt of channel x is mapped to low-level interrupt line. Write: Leaves the bit and the corresponding set bit in the ESMILSR1 register unchanged.
		1	Read: Interrupt of channel x is mapped to high-level interrupt line. Write: Maps interrupt of channel x to low-level interrupt line and clears the corresponding set bit in the ESMILSR1 register.

11.4.7 ESM Status Register 1 (ESMSR1)

This register is dedicated for Group1. Note that the ESMSR1 status register will get updated if an error condition occurs, regardless if the corresponding interrupt enable flag is set or not.

Figure 11-17. ESM Status Register 1 (ESMSR1) [address = FFFF F518h]

31	16
ESF	
R/W1CP-X/0	

15	0
ESF	
R/W1CP-X/0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 11-9. ESM Status Register 1 (ESMSR1) Field Descriptions

Bit	Field	Value	Description
31-0	ESF		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred; no interrupt is pending. Write: Leaves the bit unchanged.
		1	Read: Error occurred; interrupt is pending. Write: Clears the bit. **Note:** After $\overline{\text{RST}}$, if one of these flags are set and the corresponding interrupt are enabled, the interrupt service routine will be called.

11.4.8 ESM Status Register 2 (ESMSR2)

This register is dedicated for Group2.

Figure 11-18. ESM Status Register 2 (ESMSR2) [address = FFFF F51Ch]

31	16
ESF	
R/W1CP-0	

15	0
ESF	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 11-10. ESM Status Register 2 (ESMSR2) Field Descriptions

Bit	Field	Value	Description
31-0	ESF		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred; no interrupt is pending. Write: Leaves the bit unchanged.
		1	Read: Error occurred; interrupt is pending. Write: Clears the bit. ESMSSR2 is not impacted by this action. **Note:** In normal operation, the flag gets cleared when reading the appropriate vector in the ESMIOFFHR offset register. Reading ESMIOFFHR will not clear the ESMSR1 and the shadow register ESMSSR2.

11.4.9 ESM Status Register 3 (ESMSR3)

This register is dedicated for Group3.

Figure 11-19. ESM Status Register 3 (ESMSR3)
[address = FFFF F520h]

31	16
ESF	
R/W1CP-X/0	

15	0
ESF	
R/W1CP-X/0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 11-11. ESM Status Register 3 (ESMSR3) Field Descriptions

Bit	Field	Value	Description
31-0	ESF		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred. Write: Leaves the bit unchanged.
		1	Read: Error occurred. Write: Clears the bit.

11.4.10 ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR)

Figure 11-20. ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR)
[address = FFFF F524h]

31	16
Reserved	
R-0	

15	1	0
Reserved		EPSF
R-0		R-X/1

LEGEND: R = Read only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 11-12. ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	EPSF		$\overline{\text{ERROR}}$ Pin Status Flag. Provides status information for the $\overline{\text{ERROR}}$ pin. **Read/Write in User and Privileged mode.**
		0	Read: $\overline{\text{ERROR}}$ pin is low (active) if any error has occurred. Write: Writes have no effect.
		1	Read: $\overline{\text{ERROR}}$ pin is high if no error has occurred. Write: Writes have no effect. **Note:** This flag will be set to 1 after $\overline{\text{PORRST}}$. The value will be unchanged after $\overline{\text{RST}}$. The $\overline{\text{ERROR}}$ pin status remains unchanged after $\overline{\text{RST}}$.

11.4.11 ESM Interrupt Offset High Register (ESMIOFFHR)

Figure 11-21. ESM Interrupt Offset High Register (ESMIOFFHR) [address = FFFF F528h]

31	16
Reserved	
R-0	

15	7	6	0
Reserved		INTOFFH	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 11-13. ESM Interrupt Offset High Register (ESMIOFFHR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reads return 0. Writes have no effect.
6-0	INTOFFH		Offset High-Level Interrupt. This vector gives the channel number of the highest-pending interrupt request for the high-level interrupt line. Interrupts of error Group2 have higher priority than interrupts of error Group1. Inside a group, channel 0 has highest priority and channel 31 has lowest priority.
			User and privileged mode (read):
			Returns number of pending interrupt with the highest priority for the high-level interrupt line.
		0	No pending interrupt.
		1h	Interrupt pending for channel 0, error Group1.
		:	:
		20h	Interrupt pending for channel 31, error Group1.
		21h	Interrupt pending for channel 0, error Group2.
		:	:
		40h	Interrupt pending for channel 31, error Group2.
		41h	Interrupt pending for channel 32, error Group1.
		:	:
		60h	Interrupt pending for channel 63, error Group1.
			Note: Reading the interrupt vector will clear the corresponding flag in the ESMSR2 register; will **not** clear ESMSR1 and ESMSSR2 and the offset register gets updated.
			User and privileged mode (write):
			Writes have no effect.

11.4.12 ESM Interrupt Offset Low Register (ESMIOFFLR)

Figure 11-22. ESM Interrupt Offset Low Register (ESMIOFFLR) [address = FFFF F52Ch]

31	16
Reserved	
R-0	

15	7	6	0
Reserved		INTOFFL	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 11-14. ESM Interrupt Offset Low Register (ESMIOFFLR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reads return 0. Writes have no effect.
6-0	INTOFFL		Offset Low-Level Interrupt. This vector gives the channel number of the highest-pending interrupt request for the low-level interrupt line. Inside a group, channel 0 has highest priority and channel 31 has lowest priority.
			User and privileged mode (read):
			Returns number of pending interrupt with the highest priority for the low-level interrupt line.
		0	No pending interrupt.
		1h	Interrupt pending for channel 0, error Group1.
		:	:
		20h	Interrupt pending for channel 31, error Group1.
		21h-40h	Reserved
		41h	Interrupt pending for channel 32, error Group1.
		:	:
		60h	Interrupt pending for channel 63, error Group1.
			Note: Reading the interrupt vector will **not** clear the corresponding flag in the ESMSR1 register. Group2 interrupts are fixed to the high-level interrupt line only.
			User and privileged mode (write):
			Writes have no effect.

11.4.13 ESM Low-Time Counter Register (ESMLTCR)

Figure 11-23. ESM Low-Time Counter Register (ESMLTCR) [address = FFFF F530h]

31	16
Reserved	
R-0	

15	0
LTC	
R-3FFFh	

LEGEND: R = Read only; -*n* = value after reset

Table 11-15. ESM Low-Time Counter Register (ESMLTCR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	LTC		ERROR Pin Low-Time Counter 16-bit preloadable down-counter to control low-time of $\overline{\text{ERROR}}$ pin. The low-time counter is triggered by the peripheral clock (VCLK). **Note:** Low-time counter is set to the default preload value of the ESMLTCPR in the following cases: 1. Reset (power on reset or warm reset) 2. An error occurs 3. User forces an error

11.4.14 ESM Low-Time Counter Preload Register (ESMLTCPR)

Figure 11-24. ESM Low-Time Counter Preload Register (ESMLTCPR) [address = FFFF F534h]

31	16
Reserved	
R-0	

15	14	13	0
LTCP		LTCP	
R/WP-0		R-3FFFh	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 11-16. ESM Low-Time Counter Preload Register (ESMLTCPR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	LTCP	0-FFFFh	$\overline{\text{ERROR}}$ Pin Low-Time Counter Pre-load Value 16-bit preload value for the $\overline{\text{ERROR}}$ pin low-time counter. Defines the minimum period for which the $\overline{\text{ERROR}}$ pin will be driven to 16384 VCLK cycles. **Note:** Only LTCP[15] and LTCP[14] are configurable (privileged mode write).

11.4.15 ESM Error Key Register (ESMEKR)

Figure 11-25. ESM Error Key Register (ESMEKR)
[address = FFFF F538h]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		EKEY	
R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 11-17. ESM Error Key Register (ESMEKR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EKEY		Error Key. The key to reset the $\overline{\text{ERROR}}$ pin or to force an error on the $\overline{\text{ERROR}}$ pin.
			User and privileged mode (read):
			Returns current value of the EKEY.
			Privileged mode (write):
		0	Activates normal mode (recommended default mode).
		5h	The $\overline{\text{ERROR}}$ pin set to high when the low-time counter (LTC) has completed; then the EKEY bit will switch back to normal mode (EKEY = 0000).
		Ah	Forces error on $\overline{\text{ERROR}}$ pin.
		All other values	Activates normal mode.

11.4.16 ESM Status Shadow Register 2 (ESMSSR2)

This register is dedicated for Group2.

Figure 11-26. ESM Status Shadow Register 2 (ESMSSR2)
[address = FFFF F53Ch]

31	16
ESF	
R/W1CP-X/0	

15	0
ESF	
R/W1CP-X/0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 11-18. ESM Status Shadow Register 2 (ESMSSR2) Field Descriptions

Bit	Field	Value	Description
31-0	ESF		Error Status Flag. Shadow register for status information on pending error.
			Read in User and Privileged mode. Write in Privileged mode only.
		0	Read: No error occurred.
			Write: Leaves the bit unchanged.
		1	Read: Error occurred.
			Write: Clears the bit. ESMSR2 is not impacted by this action.
			Note: Errors are stored until they are cleared by the software or at power-on reset ($\overline{\text{PORRST}}$).

11.4.17 ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4)

This register is dedicated for Group1.

Figure 11-27. ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4) [address = FFFF F540h]

31	16
IEPSET[63:48]	
R/WP-0	
15	0
IEPSET[47:32]	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; *-n* = value after reset

Table 11-19. ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4) Field Descriptions

Bit	Field	Value	Description
63-32	IEPSET		Set influence on $\overline{\text{ERROR}}$ pin. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. Write: Leaves the bit and the corresponding clear bit in the ESMIEPCR4 register unchanged.
		1	Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. Write: Enables failure influence on $\overline{\text{ERROR}}$ pin and sets the corresponding clear bit in the ESMIEPCR4 register.

11.4.18 ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4)

This register is dedicated for Group1.

Figure 11-28. ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4) [address = FFFF F544h]

31	16
IEPCLR[63:48]	
R/WP-0	
15	0
IEPCLR[47:32]	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; *-n* = value after reset

Table 11-20. ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4) Field Descriptions

Bit	Field	Value	Description
63-32	IEPCLR		Clear influence on $\overline{\text{ERROR}}$ pin. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. Write: Leaves the bit and the corresponding set bit in the ESMIEPSR4 register unchanged.
		1	Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. Write: Disables failure influence on $\overline{\text{ERROR}}$ pin and clears the corresponding set bit in the ESMIEPSR4 register.

11.4.19 ESM Interrupt Enable Set Register 4 (ESMIESR4)

This register is dedicated for Group1.

Figure 11-29. ESM Interrupt Enable Set Register 4 (ESMIESR4) [address = FFFF F548h]

31	16
INTENSET[63:48]	
R/WP-0	
15	0
INTENSET[47:32]	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 11-21. ESM Interrupt Enable Set Register 4 (ESMIESR4) Field Descriptions

Bit	Field	Value	Description
63-32	INTENSET		Set interrupt enable.
			Read in User and Privileged mode. Write in Privileged mode only.
		0	Read: Interrupt is disabled.
			Write: Leaves the bit and the corresponding clear bit in the ESMIECR4 register unchanged.
		1	Read: Interrupt is enabled.
			Write: Enables interrupt and sets the corresponding clear bit in the ESMIECR4 register.

11.4.20 ESM Interrupt Enable Clear Register 4 (ESMIECR4)

This register is dedicated for Group1.

Figure 11-30. ESM Interrupt Enable Clear Register 4 (ESMIECR4) [address = FFFF F54Ch]

31	16
INTENCLR[63:48]	
R/WP-0	
15	0
INTENCLR[47:32]	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 11-22. ESM Interrupt Enable Clear Register 4 (ESMIECR4) Field Descriptions

Bit	Field	Value	Description
63-32	INTENCLR		Clear interrupt enable.
			Read in User and Privileged mode. Write in Privileged mode only.
		0	Read: Interrupt is disabled.
			Write: Leaves the bit and the corresponding set bit in the ESMIESR4 register unchanged.
		1	Read: Interrupt is enabled.
			Write: Disables interrupt and clears the corresponding set bit in the ESMIESR4 register.

11.4.21 ESM Interrupt Level Set Register 4 (ESMILSR4)

This register is dedicated for Group1.

Figure 11-31. ESM Interrupt Level Set Register 4 (ESMILSR4) [address = FFFF F550h]

31	16
INTLVLSET[63:48]	
R/WP-0	
15	0
INTLVLSET[47:32]	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; *-n* = value after reset

Table 11-23. ESM Interrupt Level Set Register 4 (ESMILSR4) Field Descriptions

Bit	Field	Value	Description
63-32	INTLVLSET		Set interrupt level. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Read: Interrupt of channel x is mapped to low-level interrupt line. Write: Leaves the bit and the corresponding clear bit in the ESMILCR4 register unchanged.
		1	Read: Interrupt of channel x is mapped to high-level interrupt line. Write: Maps interrupt of channel x to high-level interrupt line and sets the corresponding clear bit in the ESMILCR4 register.

11.4.22 ESM Interrupt Level Clear Register 4 (ESMILCR4)

This register is dedicated for Group1.

Figure 11-32. ESM Interrupt Level Clear Register 4 (ESMILCR4) [address = FFFF F554h]

31	16
INTLVLCLR[63:48]	
R/WP-0	
15	0
INTLVLCLR[47:32]	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; *-n* = value after reset

Table 11-24. ESM Interrupt Level Clear Register 4 (ESMILCR4) Field Descriptions

Bit	Field	Value	Description
63-32	INTLVLCLR		Clear interrupt level. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Interrupt of channel x is mapped to low-level interrupt line. Write: Leaves the bit and the corresponding set bit in the ESMILSR4 register unchanged.
		1	Read: Interrupt of channel x is mapped to high-level interrupt line. Write: Maps interrupt of channel x to low-level interrupt line and clears the corresponding set bit in the ESMILSR4 register.

11.4.23 ESM Status Register 4 (ESMSR4)

This register is dedicated for Group1.

Figure 11-33. ESM Status Register 4 (ESMSR4) [address = FFFF F558h]

31	16
ESF[63:48]	
R/W1CP-X/0	
15	**0**
ESF[47:32]	
R/W1CP-X/0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 11-25. ESM Status Register 4 (ESMSR4) Field Descriptions

Bit	Field	Value	Description
63-32	ESF		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred; no interrupt is pending. Write: Leaves the bit unchanged.
		1	Read: Error occurred; interrupt is pending. Write: Clears the bit. **Note:** After $\overline{\text{RST}}$, if one of these flags are set and the corresponding interrupt are enabled, the interrupt service routine will be called.

Real-Time Interrupt (RTI) Module

This chapter describes the functionality of the real-time interrupt (RTI) module. The RTI is designed as an operating system timer to support a real time operating system (RTOS).

Topic **Page**

12.1 Overview

The real-time interrupt (RTI) module provides timer functionality for operating systems and for benchmarking code. The RTI module can incorporate several counters that define the time bases needed for scheduling in the operating system.

The timers also allow you to benchmark certain areas of code by reading the values of the counters at the beginning and the end of the desired code range and calculating the difference between the values.

12.1.1 Features

The RTI module has the following features:

- Two independent 64 bit counter blocks
- Four configurable compares for generating operating system ticks. Each event can be driven by either counter block 0 or counter block 1.
- Fast enabling/disabling of events
- Two time stamp (capture) functions for system or peripheral interrupts, one for each counter block
- Digital windowed watchdog

12.1.2 Industry Standard Compliance Statement

This module is specifically designed to fulfill the requirements for OSEK (**O**ffene **S**ysteme und deren **S**chnittstellen für die **E**lektronik im **K**raftfahrzeug, or Open Systems and the Corresponding Interfaces for Automotive Electronics) as well as OSEK/time-compliant operating systems, but is not limited to it.

12.2 Module Operation

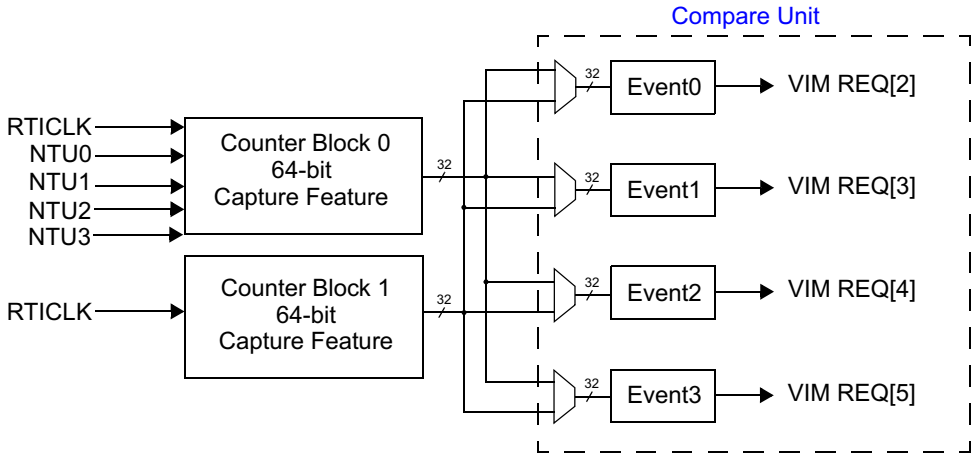
Figure 12-1 illustrates the high level block diagram of the RTI module.

The RTI module has two independent counter blocks for generating different time bases: counter block 0 and counter block 1.

A compare unit compares the counters with programmable values and generates four independent interrupt on compare matches. Each of the compare registers can be programmed to be compared to either counter block 0 or counter block 1.

The following sections describe the individual functions in more detail.

Figure 12-1. RTI Block Diagram



12.2.1 Counter Operation

Each counter block consists of the following (see Figure 12-2):

- One 32-bit prescale counter (RTIUC0 or RTIUC1)
- One 32-bit free running counter (RTIFRC0 or RTIFRC1)

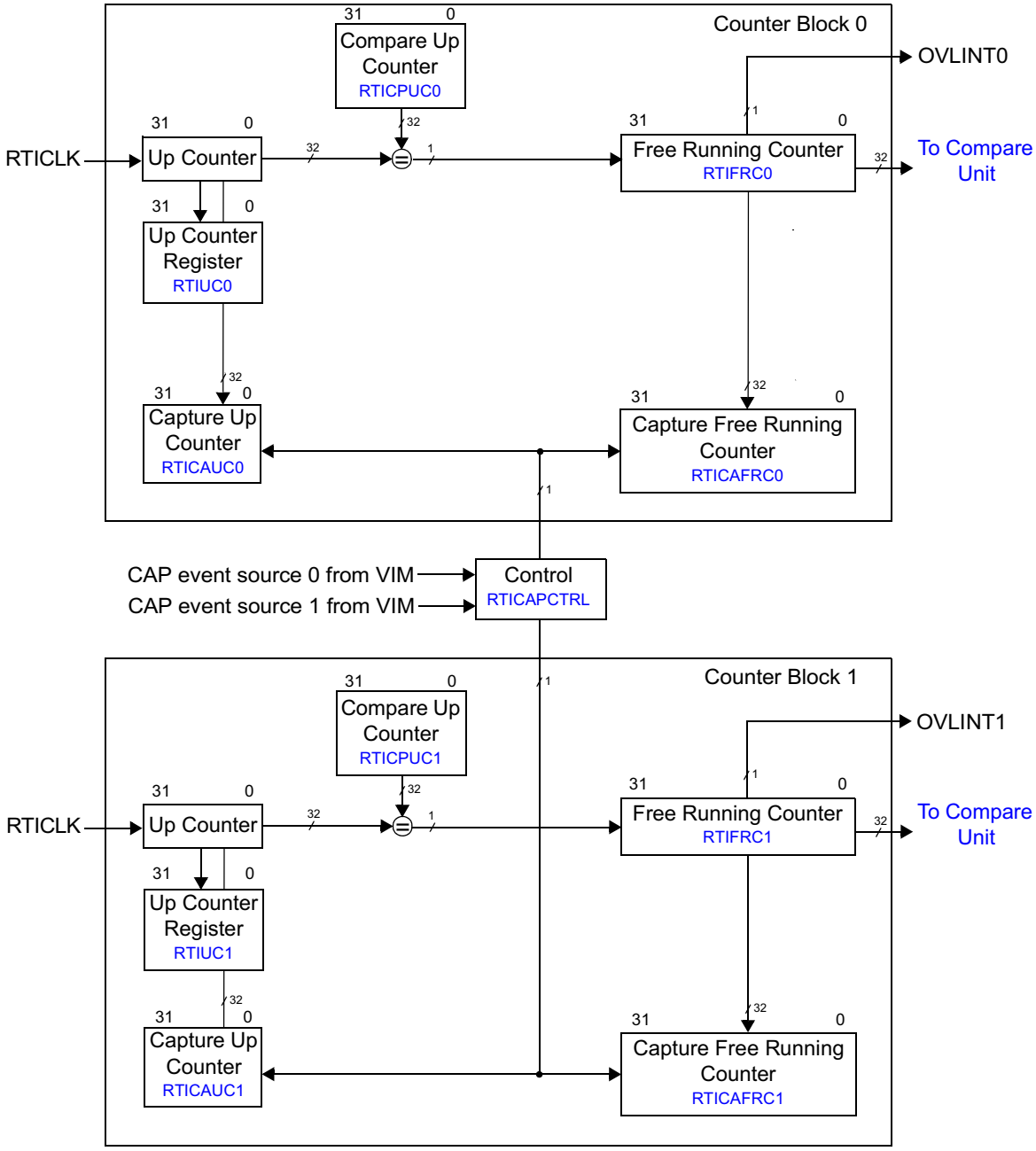
The RTIUC0/1 is driven by the RTICLK and counts up until the compare value in the compare up counter register (RTICPUC0 or RTICPUC1) is reached. When the compare matches, RTIFRC0/1 is incremented and RTIUC0/1 is reset to 0. If RTIFRC0/1 overflows, an interrupt is generated to the vectored interrupt manager (VIM). The overflow interrupt is not intended to generate the time base for the operating system. See Section 12.2.2 for the time base generation. The up counter together with the compare up counter value prescale the RTI clock. The resulting formula for the frequency of the free running counter (RTIFRC0/1) is:

$$f_{RTIFRCx} = \begin{cases} \dfrac{f_{RTICLK}}{RTICPUCx + 1} & \text{when RTICPUCx} \neq 0 \\ \dfrac{f_{RTICLK}}{2^{32}} & \text{when RTICPUCx} = 0 \end{cases} \quad (23)$$

NOTE: Setting RTICPUCx equal to zero is not recommended. Doing so will hold the Up Counter at zero for two RTICLK cycles after it overflows from 0xFFFFFFFF to zero.

The counter values can be determined by reading the respective counter registers or by generating a hardware event that captures the counter value into the respective capture register. Both functions are described in the following sections.

Figure 12-2. Counter Block Diagram



12.2.1.1 Counter and Capture Read Consistency

Portions of the device internal databus are 32-bits wide. If the application wants to read the 64-bit counters or the 64-bit capture values, a certain order of 32-bit read operations needs to be followed. This is to prevent one counter incrementing in between the two separate read operations to both counters.

Reading the Counters

The free running counter (RTIFRCx) must be read first. This priority will ensure that in the cycle when the CPU reads RTIFRCx, the up counter value is stored in its counter register (RTIUCx). The second read has to access the up counter register (RTIUCx), which then holds the value which corresponds to the number of RTICLK cycles that have elapsed at the time reading the free running counter register (RTIFRCx).

NOTE: The up counters are implemented as shadow registers. Reading RTIUCx without having read RTIFRCx first will return always the same value. RTIUCx will only be updated when RTIFRCx is read.

Reading the Capture Values

The free running counter capture register (RTICAFRCx) must be read first. This priority will ensure that in the cycle when the CPU reads RTICAFRCx, the up counter value is stored in its counter register (RTICAUCx). The second read has to access the up counter register (RTICAUCx), which then holds the value captured at the time when reading the capture free running counter register (RTICAFRCx).

NOTE: The capture up counter registers are implemented as shadow registers. Reading RTICAUCx without having read RTICAFRCx first will return always the same value. RTICAUCx will only be updated when RTICAFRCx is read.

12.2.1.2 Capture Feature

Both counter blocks also provide a capture feature on external events. Two capture sources can trigger the capture event. The source triggering the block is configurable (RTICAPCTRL). The sources originate from the Vectored Interrupt Manager (VIM) and allow the generation of capture events when a peripheral modules has generated an interrupt. Any of the peripheral interrupts can be selected as the capture event in the VIM.

When an event is detected, RTIUCx and RTIFRCx are stored in the capture up counter (RTICAUCx) and capture free running counter (RTICAFRCx) registers. The read order of the captured values must be the same as the read order of the actual counters (see Section 12.2.1.1).

12.2.2 Interrupt Requests

There are four compare registers (RTICOMPy) to generate interrupt requests to the VIM. The interrupts can be used to generate different time bases for the operating system. Each of the compare registers can be configured to be compared to either RTIFRC0 or RTIFRC1. When the counter value matches the compare value, an interrupt is generated. To allow periodic interrupts, a certain value can be added to the compare value in RTICOMPy automatically. This value is stored in the update compare register (RTIUDCPy) and will be added after a compare is matched. The period of the generated interrupt can be calculated with:

$$t_{COMPx} = t_{RTICLK} \times (RTICPUCy + 1) \times RTIUDCPy$$

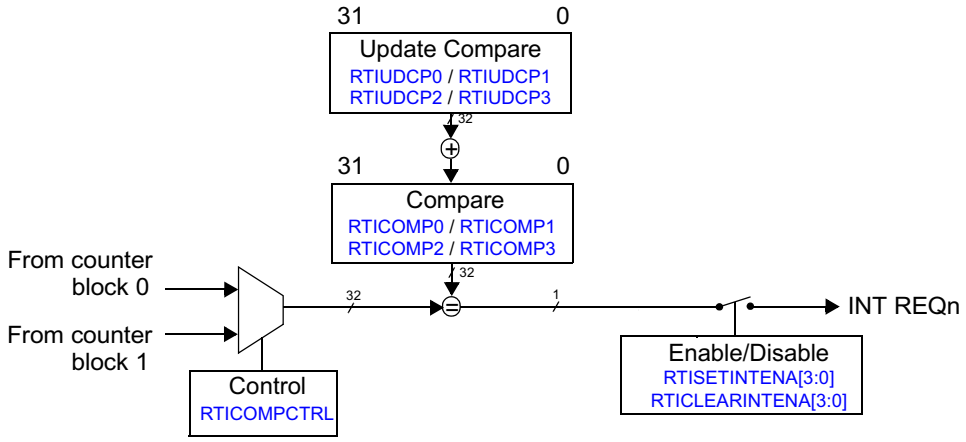
if RTICPUCy ≠ 0,

$$t_{COMPx} = t_{RTICLK} \times 2^{32} \times RTIUDCPy$$

if RTIUDCPy = 0,

$$t_{COMPx} = t_{RTICLK} \times (RTICPUCy + 1) \times 2^{32} \quad (24)$$

Figure 12-3. Compare Unit Block Diagram (shows only 1 of 4 blocks for simplification)



Another interrupt that can be generated is the overflow interrupt (OVLINTx) in case the RTIFRCx counter overflows.

The interrupts can be enabled in the RTISETINTENA register and disabled in the RTICLEARINTENA register. The RTIINTFLAG register shows the pending interrupts.

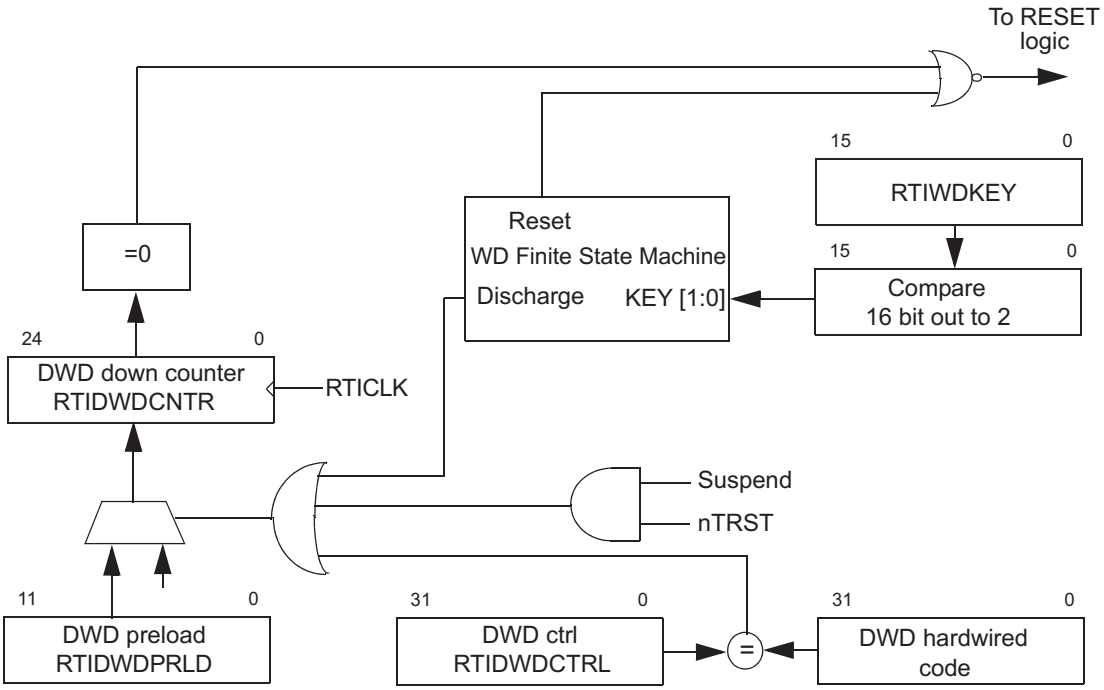
12.2.3 RTI Clocking

The counter blocks are clocked with RTICLK (for definition, see Section 2.4.2).

12.2.4 Digital Watchdog (DWD)

The digital watchdog (DWD) is an optional safety diagnostic which can detect a runaway CPU and generate either a reset or NMI (non-maskable interrupt) response. It generates resets or NMIs after a programmable period, or if no correct key sequence was written to the RTIWDKEY register. Figure 12-4 illustrates the DWD.

Figure 12-4. Digital Watchdog



12.2.4.1 Digital Watchdog (DWD)

The DWD is disabled by default. If it should be used, it must be enabled by writing a 32-bit value to the RTIDWDCTRL register.

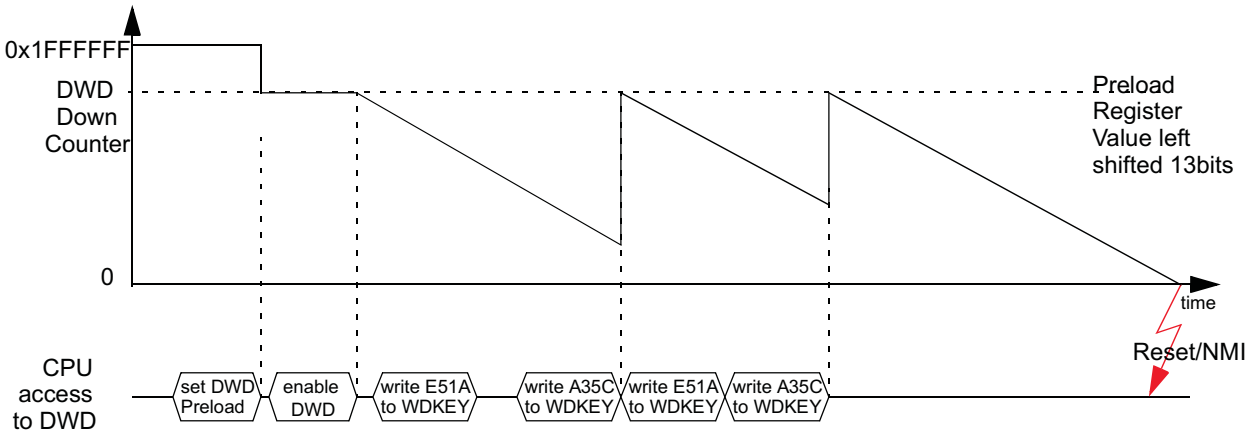
NOTE: Once the DWD is enabled, it cannot be disabled except by system reset or power on reset.

If the correct key sequence is written to the RTIWDKEY register (0xE51A followed by 0xA35C), the 25-bit DWD down counter is reloaded with the left justified 12-bit preload value stored in RTIDWDPRLD. If an incorrect value is written, a watchdog reset or NMI will occur immediately. A reset or NMI will also be generated when the DWD down counter is decremented to 0.

While the device is in suspend mode (halting debug mode), the DWD down counter keeps the value it had when entering suspend mode.

The DWD down counter will be decremented with the RTICLK frequency.

Figure 12-5. DWD Operation



The expiration time of the DWD down counter can be determined with the following equation:

$$texp = (DWDPRLD + 1) \times 2^{13}/RTICLK$$

where

DWDPRLD = 0...4095

NOTE: Care should be taken to ensure that the CPU write to the watchdog register is made allowing time for the write to propagate to the RTI.

12.2.4.2 Digital Windowed Watchdog (DWWD)

In addition to the time-out boundary configurable via the digital watchdog discussed in Section 12.2.4.1, for enhanced safety metrics it is desirable to check for a watchdog "pet" within a time window rather than using a single time threshold. This is enabled by the digital windowed watchdog (DWWD) feature.

- Functional Behavior

The DWWD opens a configurable time window in which the watchdog must be serviced. Any attempt to service the watchdog outside this time window, or a failure to service the watchdog in this time window, will cause the watchdog to generate either a reset or a NMI to the CPU. This is controlled by configuring the RTIWWDRXNCTRL register. As with the DWD, the DWWD is disabled after power on reset. When the DWWD is configured to generate a non-maskable interrupt on a window violation, the watchdog counter continues to count down. The NMI handler needs to clear the watchdog violation status flag(s) and then

service the watchdog by writing the correct sequence in the watchdog key register. This service will cause the watchdog counter to get reloaded from the preload value and start counting down. If the NMI handler does not service the watchdog in time, it could count down all the way to zero and wrap around. If the NMI Handler does not service the watchdog in time, the NMI gets generated continuously, each time the counter counts to '0'.

The DWWD uses the Digital Watchdog (DWD) preload register (RTIDWDPRLD) setting to define the end-time of the window. The start-time of the window is defined by a window size configuration register (RTIWWDSIZECTRL).

The default window size is set to 100%, which corresponds to the DWD functionality of a time-out-only watchdog. The window size can be selected (through register RTIWWDSIZECTRL) from among 100%, 50%, 25%, 12.5%, 6.25% and 3.125% as shown in Figure 12-6. The window with the respective size will be opened before the end of the DWD expiration. The user has to serve the watchdog in the window. Otherwise, a reset or NMI will generate. Figure 12-7 shows an DWWD operation example (25% window).

- Configuration of DWWD

The DWWD preload value (same as DWD preload) can only be configured when the DWWD counter is disabled. The window size and watchdog reaction to a violation can be configured even after the watchdog has been enabled. Any changes to the window size and watchdog reaction configurations will only take effect after the next servicing of the DWWD. This feature can be utilized to dynamically set windows of different sizes based on task execution time, adding a program sequence element to the diagnostic which can improve fault coverage.

Figure 12-6. Digital Windowed Watchdog Timing Example

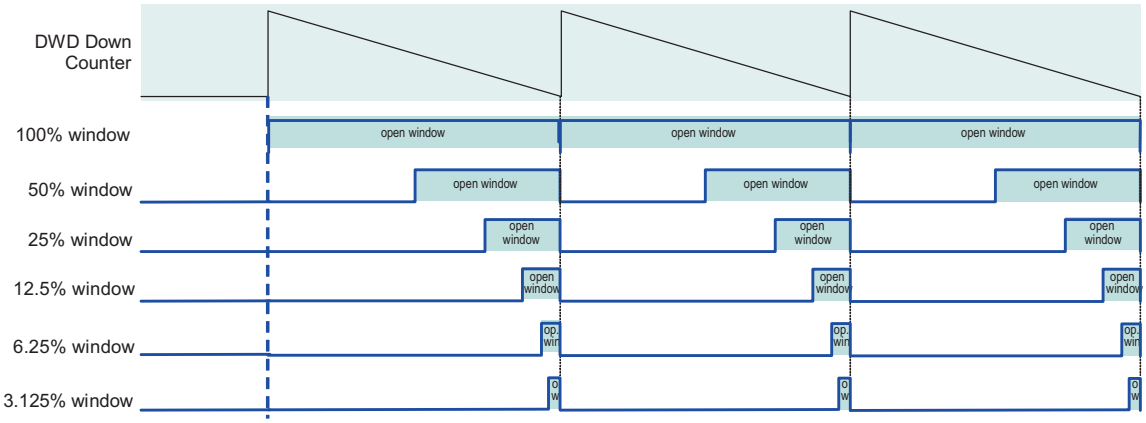
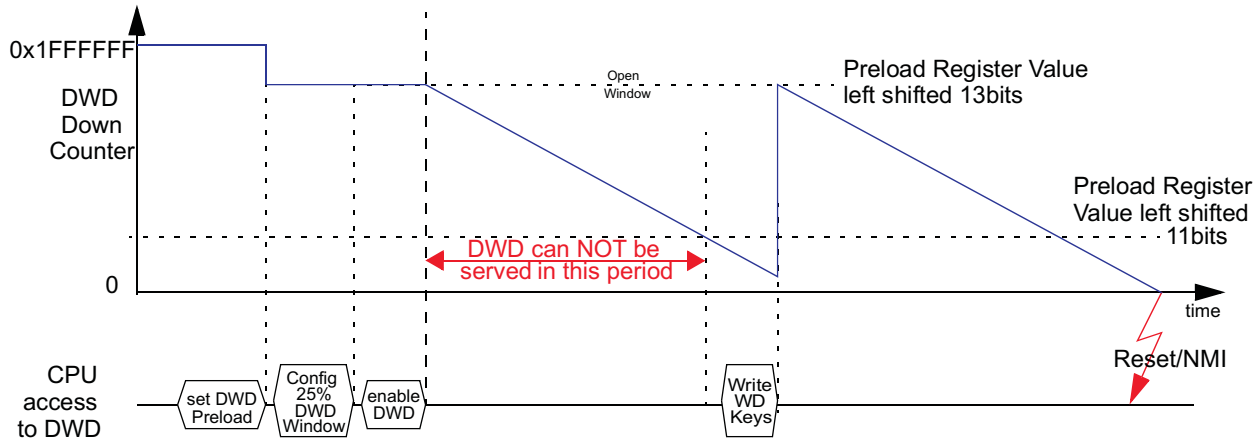


Figure 12-7. Digital Windowed Watchdog Operation Example (25% Window)



12.2.5 Low Power Modes

Low power modes allow the trade off of the current used during low power versus functionality and fast wakeup response. All low power modes have the following characteristics:

- CPU and system clocks are disabled.
- Flash banks and pump are in sleep mode.
- All peripheral modules are in low power modes and the clocks are disabled (exceptions to this may occur and would be documented in the specific device data sheet).

Flexibility in enabling and disabling clocks allows for many different low-power modes (see Section 2.4.3).

The operation of the RTI Module is assured in Run, Doze, and Snooze modes. In Sleep mode, all clocks will be switched off and the RTI will not work.

In Doze and Snooze modes, the RTI is active and is able to wake up the device with compare, time base and overflow interrupts. The compare interrupts can be used to periodically wake up the device. The overflow interrupt can be used to notify the operating system that a counter overflow has occurred. Capturing events generated by the Vectored Interrupt Module (VIM) is also possible since, in both of these low power modes, the peripheral modules are able to generate interrupts that can trigger capture events. Capturing events while in Sleep mode is not supported as the clock to the RTI is not active.

NOTE: RTICLK in Doze Mode

In the special case of Doze Mode with PLL off, RTICLK might have a different period than with PLL enabled since RTICLK will be derived from the oscillator output. It has to be ensured that the VCLK to RTICLK ratio is at least 3:1.

12.2.6 Halting Debug Mode Behaviour

Once the system enters halting debug mode, the behavior of the RTI depends on the COS (continue on suspend) bit. If the bit is cleared and halting debug mode is active, all counters will stop operation. If the bit is set to one, all counters will be clocked normally and the RTI will work like in normal mode.

12.3 RTI Control Registers

Table 12-1 provides a summary of the registers. The registers support 8-bit, 16-bit, and 32-bit writes. The offset is relative to the associated peripheral select. See the following sections for detailed descriptions of the registers. The base address for the control registers is FFFF FC00h. The address locations not listed are reserved.

Table 12-1. RTI Registers

Offset	Acronym	Register Description	Section
00h	RTIGCTRL	RTI Global Control Register	Section 12.3.1
04h	RTITBCTRL	Reserved. Do NOT use.	
08h	RTICAPCTRL	RTI Capture Control Register	Section 12.3.2
0Ch	RTICOMPCTRL	RTI Compare Control Register	Section 12.3.3
10h	RTIFRC0	RTI Free Running Counter 0 Register	Section 12.3.4
14h	RTIUC0	RTI Up Counter 0 Register	Section 12.3.5
18h	RTICPUC0	RTI Compare Up Counter 0 Register	Section 12.3.6
20h	RTICAFRC0	RTI Capture Free Running Counter 0 Register	Section 12.3.7
24h	RTICAUC0	RTI Capture Up Counter 0 Register	Section 12.3.8
30h	RTIFRC1	RTI Free Running Counter 1 Register	Section 12.3.9
34h	RTIUC1	RTI Up Counter 1 Register	Section 12.3.10
38h	RTICPUC1	RTI Compare Up Counter 1 Register	Section 12.3.11
40h	RTICAFRC1	RTI Capture Free Running Counter 1 Register	Section 12.3.12
44h	RTICAUC1	RTI Capture Up Counter 1 Register	Section 12.3.13
50h	RTICOMP0	RTI Compare 0 Register	Section 12.3.14
54h	RTIUDCP0	RTI Update Compare 0 Register	Section 12.3.15
58h	RTICOMP1	RTI Compare 1 Register	Section 12.3.16
5Ch	RTIUDCP1	RTI Update Compare 1 Register	Section 12.3.17
60h	RTICOMP2	RTI Compare 2 Register	Section 12.3.18
64h	RTIUDCP2	RTI Update Compare 2 Register	Section 12.3.19
68h	RTICOMP3	RTI Compare 3 Register	Section 12.3.20
6Ch	RTIUDCP3	RTI Update Compare 3 Register	Section 12.3.21
70h	RTITBLCOMP	Reserved. Do not use.	-
74h	RTITBHCOMP	Reserved. Do not use.	-
80h	RTISETINTENA	RTI Set Interrupt Enable Register	Section 12.3.22
84h	RTICLEARINTENA	RTI Clear Interrupt Enable Register	Section 12.3.23
88h	RTIINTFLAG	RTI Interrupt Flag Register	Section 12.3.24
90h	RTIDWDCTRL	Digital Watchdog Control Register	Section 12.3.25
94h	RTIDWDPRLD	Digital Watchdog Preload Register	Section 12.3.26
98h	RTIWDSTATUS	Watchdog Status Register	Section 12.3.27
9Ch	RTIWDKEY	RTI Watchdog Key Register	Section 12.3.28
A0h	RTIDWDCNTR	RTI Digital Watchdog Down Counter Register	Section 12.3.29
A4h	RTIWWDRXNCTRL	Digital Windowed Watchdog Reaction Control Register	Section 12.3.30
A8h	RTIWWDSIZECTRL	Digital Windowed Watchdog Window Size Control Register	Section 12.3.31
ACh	RTIINTCLRENABLE	RTI Compare Interrupt Clear Enable Register	Section 12.3.32
B0h	RTICOMP0CLR	RTI Compare 0 Clear Register	Section 12.3.33
B4h	RTICOMP1CLR	RTI Compare 1 Clear Register	Section 12.3.34
B8h	RTICOMP2CLR	RTI Compare 2 Clear Register	Section 12.3.35
BCh	RTICOMP3CLR	RTI Compare 3 Clear Register	Section 12.3.36

NOTE: Writes to Reserved registers may clear the pending RTI interrupt.

12.3.1 RTI Global Control Register (RTIGCTRL)

The global control register starts/stops the counters and selects the signal compared with the timebase control circuit. This register is shown in Figure 12-8 and described in Table 12-2.

Figure 12-8. RTI Global Control Register (RTIGCTRL) [offset = 00]

31	20	19	16
Reserved		Reserved	
R-0		R/WP-0	

15	14	2	1	0
COS	Reserved		CNT1EN	CNT0EN
R/WP-0	R-0		R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-2. RTI Global Control Register (RTIGCTRL) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	Reserved	0	Reserved. Do NOT use.
15	COS		Continue on suspend. This bit determines if both counters are stopped when the device goes into halting debug mode or if they continue counting.
		0	Counters are stopped while in halting debug mode.
		1	Counters are running while in halting debug mode.
14-2	Reserved	0	Reads return 0. Writes have no effect.
1	CNT1EN		Counter 1 enable. This bit starts and stops counter block 1 (RTIUC1 and RTIFRC1).
		0	Counter block 1 is stopped.
		1	Counter block 1 is running.
0	CNT0EN		Counter 0 enable. This bit starts and stops counter block 0 (RTIUC0 and RTIFRC0).
		0	Counter block 0 is stopped.
		1	Counter block 0 is running.

12.3.2 RTI Capture Control Register (RTICAPCTRL)

The capture control register controls the capture source for the counters. This register is shown in Figure 12-9 and described in Table 12-3.

Figure 12-9. RTI Capture Control Register (RTICAPCTRL) [offset = 08h]

31	8
Reserved	
R-0	

7	2	1	0
Reserved		CAPCNTR1	CAPCNTR0
R-0		R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-3. RTI Capture Control Register (RTICAPCTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	CAPCNTR1		Capture counter 1. This bit determines which external interrupt source triggers a capture event of RTIUC1 and RTIFRC1.
		0	Capture of RTIUC1/ RTIFRC1 is triggered by capture event source 0.
		1	Capture of RTIUC1/ RTIFRC1 is triggered by capture event source 1.
0	CAPCNTR0		Capture counter 0. This bit determines which external interrupt source triggers a capture event of RTIUC0 and RTIFRC0.
		0	Capture of RTIUC0/ RTIFRC0 is triggered by capture event source 0.
		1	Capture of RTIUC0/ RTIFRC0 is triggered by capture event source 1.

12.3.3 RTI Compare Control Register (RTICOMPCTRL)

The compare control register controls the source for the compare registers. This register is shown in Figure 12-10 and described in Table 12-4.

Figure 12-10. RTI Compare Control Register (RTICOMPCTRL) [offset = 0Ch]

31	16
Reserved	
R-0	

15	13	12	11	9	8
Reserved		COMPSEL3	Reserved		COMPSEL2
R-0		R/WP-0	R-0		R/WP-0

7	5	4	3	1	0
Reserved		COMPSEL1	Reserved		COMPSEL0
R-0		R/WP-0	R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-4. RTI Compare Control Register (RTICOMPCTRL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12	COMPSEL3		Compare select 3. This bit determines the counter with which the compare value held in compare register 3 (RTICOMP3) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
11-9	Reserved	0	Reads return 0. Writes have no effect.
8	COMPSEL2		Compare select 2. This bit determines the counter with which the compare value held in compare register 2 (RTICOMP2) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	COMPSEL1		Compare select 1. This bit determines the counter with which the compare value held in compare register 1 (RTICOMP1) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
3-1	Reserved	0	Reads return 0. Writes have no effect.
0	COMPSEL0		Compare select 0. This bit determines the counter with which the compare value held in compare register 0 (RTICOMP0) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.

12.3.4 RTI Free Running Counter 0 Register (RTIFRC0)

The free running counter 0 register holds the current value of free running counter 0. This register is shown in Figure 12-11 and described in Table 12-5.

Figure 12-11. RTI Free Running Counter 0 Register (RTIFRC0) [offset = 10h]

31	16
FRC0	
R/WP-0	

15	0
FRC0	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-5. RTI Free Running Counter 0 Register (RTIFRC0) Field Descriptions

Bit	Field	Value	Description
31-0	FRC0	0-FFFF FFFFh	Free running counter 0. This registers holds the current value of the free running counter 0. A read of this counter returns the current value of the counter. The counter can be preset by writing (in privileged mode only) to this register. The counter increments then from this written value upwards. **Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0.**

12.3.5 RTI Up Counter 0 Register (RTIUC0)

The up counter 0 register holds the current value of prescale counter. This register is shown in Figure 12-12 and described in Table 12-6.

Figure 12-12. RTI Up Counter 0 Register (RTIUC0) [offset = 14h]

31	16
UC0	
R/WP-0	

15	0
UC0	
R/WP-0	

LLEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-6. RTI Up Counter 0 Register (RTIUC0) Field Descriptions

Bit	Field	Value	Description
31-0	UC0	0-FFFF FFFFh	Up counter 0. This register holds the current value of the up counter 0 and prescales the RTI clock. It will be only updated by a previous read of free running counter 0 (RTIFRC0). This method of updating effectively gives a 64-bit read of both counters, without having the problem of a counter being updated between two consecutive reads on up counter 0 (RTIUC0) and free running counter 0 (RTIFRC0). A read of this counter returns the value of the counter at the time RTIFRC0 was read. A write to this counter presets it with a value. The counter then increments from this written value upwards. Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0. **Note: If the preset value is bigger than the compare value stored in register RTICPUC0, then it can take a long time until a compare matches, since RTIUC0 has to count up until it overflows.**

12.3.6 RTI Compare Up Counter 0 Register (RTICPUC0)

The compare up counter 0 register holds the value to be compared with prescale counter 0 (RTIUC0). This register is shown in Figure 12-13 and described in Table 12-7.

Figure 12-13. RTI Compare Up Counter 0 Register (RTICPUC0) [offset = 18h]

31	16
CPUC0	
R/WP-0	

15	0
CPUC0	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-7. RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions

Bit	Field	Value	Description
31-0	CPUC0	0-FFFF FFFFh	Compare up counter 0. This register holds the value that is compared with the up counter 0. When the compare shows a match, the free running counter 0 (RTIFRC0) is incremented. RTIUC0 is set to 0 when the counter value matches the RTICPUC0 value. The value set in this register prescales the RTI clock. If CPUC0 = 0, then: f_{FRC0} = RTICLK/(2^{32} + 1) (Setting CPUC0 equal to 0 is not recommended. Doing so will hold the up counter at 0 for 2 RTICLK cycles after it overflows from FFFF FFFFh to 0.) If CPUC0 ≠ 0, then: f_{FRC0} = RTICLK/(RTICPUC0 + 1) A read of this register returns the current compare value. A write to this register: • If TBEXT = 0, the compare value is updated. • If TBEXT = 1, the compare value is unchanged.

12.3.7 RTI Capture Free Running Counter 0 Register (RTICAFRC0)

The capture free running counter 0 register holds the free running counter 0 on external events. This register is shown in Figure 12-14 and described in Table 12-8.

Figure 12-14. RTI Capture Free Running Counter 0 Register (RTICAFRC0) [offset = 20h]

31	16
CAFRC0	
R-0	

15	0
CAFRC0	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 12-8. RTI Capture Free Running Counter 0 Register (RTICAFRC0) Field Descriptions

Bit	Field	Value	Description
31-0	CAFRC0	0-FFFF FFFFh	Capture free running counter 0. This register captures the current value of the free running counter 0 (RTIFRC0) when an event occurs, controlled by the external capture control block. A read of this register returns the value of RTIFRC0 on a capture event.

12.3.8 RTI Capture Up Counter 0 Register (RTICAUC0)

The capture up counter 0 register holds the current value of prescale counter 0 on external events. This register is shown in Figure 12-15 and described in Table 12-9.

Figure 12-15. RTI Capture Up Counter 0 Register (RTICAUC0) [offset = 24]

31	16
CAUC0	
R-0	

15	0
CAUC0	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 12-9. RTI Capture Up Counter 0 Register (RTICAUC0) Field Descriptions

Bit	Field	Value	Description
31-0	CAUC0	0-FFFF FFFFh	Capture up counter 0. This register captures the current value of the up counter 0 (RTIUC0) when an event occurs, controlled by the external capture control block. **Note: The read sequence must be the same as with RTIUC0 and RTIFRC0. Therefore, the RTICAFRC0 register must be read before the RTICAUC0 register is read. This sequence ensures that the value of the RTICAUC0 register is the corresponding value to the RTICAFRC0 register, even if another capture event happens in between the two reads.** A read of this register returns the value of RTIUC0 on a capture event.

12.3.9 RTI Free Running Counter 1 Register (RTIFRC1)

The free running counter 1 register holds the current value of the free running counter 1. This register is shown in Figure 12-16 and described in Table 12-10.

Figure 12-16. RTI Free Running Counter 1 Register (RTIFRC1) [offset = 30h]

31	16
FRC1	
R/WP-0	

15	0
FRC1	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-10. RTI Free Running Counter 1 Register (RTIFRC1) Field Descriptions

Bit	Field	Value	Description
31-0	FRC1	0-FFFF FFFFh	Free running counter 1. This register holds the current value of the free running counter 1 and will be updated continuously. A read of this register returns the current value of the counter. A write to this register presets the counter. The counter increments then from this written value upwards. **Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC1 and RTIFRC1.**

12.3.10 RTI Up Counter 1 Register (RTIUC1)

The up counter 1 register holds the current value of the prescale counter 1. This register is shown in Figure 12-17 and described in Table 12-11.

Figure 12-17. RTI Up Counter 1 Register (RTIUC1) [offset = 34h]

31	16
UC1	
R/WP-0	

15	0
UC1	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-11. RTI Up Counter 1 Register (RTIUC1) Field Descriptions

Bit	Field	Value	Description
31-0	UC1	0-FFFF FFFFh	Up counter 1. This register holds the current value of the up counter 1 and prescales the RTI clock. It will be only updated by a previous read of free running counter 1 (RTIFRC1). This method of updating effectively gives a 64-bit read of both counters, without having the problem of a counter being updated between two consecutive reads on RTIUC1 and RTIFRC1. A read of this register will return the value of the counter when the RTIFRC1 was read. A write to this register presets the counter. The counter then increments from this written value upwards. **Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC1 and RTIFRC1.** **Note: If the preset value is bigger than the compare value stored in register RTICPUC1, then it can take a long time until a compare matches, since RTIUC1 has to count up until it overflows.**

12.3.11 RTI Compare Up Counter 1 Register (RTICPUC1)

The compare up counter 1 register holds the value compared with prescale counter 1. This register is shown in Figure 12-18 and described in Table 12-12.

Figure 12-18. RTI Compare Up Counter 1 Register (RTICPUC1) [offset = 38h]

31	16
CPUC1	
R/WP-0	

15	0
CPUC1	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-12. RTI Compare Up Counter 1 Register (RTICPUC1) Field Descriptions

Bit	Field	Value	Description
31-0	CPUC1	0-FFFF FFFFh	Compare up counter 1. This register holds the compare value, which is compared with the up counter 1. When the compare matches, the free running counter 1 (RTIFRC1) is incremented. The up counter is cleared to zero when the counter value matches the CPUC1 value. The value set in this prescales the RTI clock according to the following formula: If CPUC1 = 0, then: f_{FRC0} = RTICLK/(2^{32} + 1) (Setting CPUC1 equal to 0 is not recommended. Doing so will hold the up counter at 0 for 2 RTICLK cycles after it overflows from FFFF FFFFh to 0.) If CPUC1 ≠ 0, then: f_{FRC1} = RTICLK/(RTICPUC1 + 1) A read of this register returns the current compare value. A write to this register updates the compare value.

12.3.12 RTI Capture Free Running Counter 1 Register (RTICAFRC1)

The capture free running counter 1 register holds the current value of free running counter 1 on external events. This register is shown in Figure 12-19 and described in Table 12-13.

Figure 12-19. RTI Capture Free Running Counter 1 Register (RTICAFRC1) [offset = 40h]

31	16
CAFRC1	
R-0	

15	0
CAFRC1	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 12-13. RTI Capture Free Running Counter 1 Register (RTICAFRC1) Field Descriptions

Bit	Field	Value	Description
31-0	CAFRC1	0-FFFF FFFFh	Capture free running counter 1. This register captures the current value of the free running counter 1 (RTIFRC1) when an event occurs, controlled by the external capture control block. A read of this register returns the value of RTIFRC1 on a capture event.

12.3.13 RTI Capture Up Counter 1 Register (RTICAUC1)

The capture up counter 1 register holds the current value of prescale counter 1 on external events. This register is shown in Figure 12-20 and described in Table 12-14.

Figure 12-20. RTI Capture Up Counter 1 Register (RTICAUC1) [offset = 44h]

31	16
CAUC1	
R-0	

15	0
CAUC1	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 12-14. RTI Capture Up Counter 1 Register (RTICAUC1) Field Descriptions

Bit	Field	Value	Description
31-0	CAUC1	0-FFFF FFFFh	Capture up counter 1. This register captures the current value of the up counter 1 (RTIUC1) when an event occurs, controlled by the external capture control block. **Note: The RTICAFRC1 register must be read before the RTICAUC1 register is read. This sequence ensures that the value of the RTICAUC1 register is the corresponding value to the RTICAFRC1 register, even if another capture event happens in between the two reads.** A read of this register returns the value of RTIUC1 on a capture event.

12.3.14 RTI Compare 0 Register (RTICOMP0)

The compare 0 register holds the value to be compared with the counters. This register is shown in Figure 12-21 and described in Table 12-15.

Figure 12-21. RTI Compare 0 Register (RTICOMP0) [offset = 50h]

31	16
COMP0	
R/WP-0	

15	0
COMP0	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-15. RTI Compare 0 Register (RTICOMP0) Field Descriptions

Bit	Field	Value	Description
31-0	COMP0	0-FFFF FFFFh	Compare 0. This registers holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches the compare value, an interrupt is flagged. A read of this register will return the current compare value. A write to this register (in privileged mode only) will update the compare register with a new compare value.

12.3.15 RTI Update Compare 0 Register (RTIUDCP0)

The update compare 0 register holds the value to be added to the compare register 0 value on a compare match. This register is shown in Figure 12-22 and described in Table 12-16.

Figure 12-22. RTI Update Compare 0 Register (RTIUDCP0) [offset = 54h]

31	16
UDCP0	
R/WP-0	

15	0
UDCP0	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-16. RTI Update Compare 0 Register (RTIUDCP0) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP0	0-FFFF FFFFh	Update compare 0. This register holds a value that is added to the value in the compare 0 (RTICOMP0) register each time a compare matches. This function allows periodic interrupts to be generated without software intervention. A read of this register will return the value to be added to the RTICOMP0 register on the next compare match. A write to this register will provide a new update value.

12.3.16 RTI Compare 1 Register (RTICOMP1)

The compare 1 register holds the value to be compared to the counters. This register is shown in Figure 12-23 and described in Table 12-17.

Figure 12-23. RTI Compare 1 Register (RTICOMP1) [offset = 58h]

31	16
COMP1	
R/WP-0	

15	0
COMP1	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-17. RTI Compare 1 Register (RTICOMP1) Field Descriptions

Bit	Field	Value	Description
31-0	COMP1	0-FFFF FFFFh	Compare 1. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. A read of this register will return the current compare value. A write to this register will update the compare register with a new compare value.

12.3.17 RTI Update Compare 1 Register (RTIUDCP1)

The update compare 1 register holds the value to be added to the compare register 1 value on a compare match. This register is shown in Figure 12-24 and described in Table 12-18.

Figure 12-24. RTI Update Compare 1 Register (RTIUDCP1) [offset = 5Ch]

31	16
UDCP1	
R/WP-0	

15	0
UDCP1	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-18. RTI Update Compare 1 Register (RTIUDCP1) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP1	0-FFFF FFFFh	Update compare 1. This register holds a value that is added to the value in the RTICOMP1 register each time a compare matches. This process allows periodic interrupts to be generated without software intervention. A read of this register will return the value to be added to the RTICOMP1 register on the next compare match. . A write to this register will provide a new update value.

12.3.18 RTI Compare 2 Register (RTICOMP2)

The compare 2 register holds the value to be compared to the counters. This register is shown in Figure 12-25 and described in Table 12-19.

Figure 12-25. RTI Compare 2 Register (RTICOMP2) [offset = 60h]

31	16
COMP2	
R/WP-0	
15	0
COMP2	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-19. RTI Compare 2 Register (RTICOMP2) Field Descriptions

Bit	Field	Value	Description
31-0	COMP2	0-FFFF FFFFh	Compare 2. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. A read of this register will return the current compare value. A write to this register (in privileged mode only) will provide a new compare value.

12.3.19 RTI Update Compare 2 Register (RTIUDCP2)

The update compare 2 register holds the value to be added to the compare register 2 value on a compare match. This register is shown in Figure 12-26 and described in Table 12-20.

Figure 12-26. RTI Update Compare 2 Register (RTIUDCP2) [offset = 64h]

31	16
UDCP2	
R/WP-0	
15	0
UDCP2	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-20. RTI Update Compare 2 Register (RTIUDCP2) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP2	0-FFFF FFFFh	Update compare 2. This register holds a value that is added to the value in the RTICOMP2 register each time a compare matches. This process makes it possible to generate periodic interrupts without software intervention. A read of this register will return the value to be added to the RTICOMP2 register on the next compare match. A write to this register will provide a new update value.

12.3.20 RTI Compare 3 Register (RTICOMP3)

The compare 3 register holds the value to be compared to the counters. This register is shown in Figure 12-27 and described in Table 12-21.

Figure 12-27. RTI Compare 3 Register (RTICOMP3) [offset = 68h]

31	16
COMP3	
R/WP-0	

15	0
COMP3	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-21. RTI Compare 3 Register (RTICOMP3) Field Descriptions

Bit	Field	Value	Description
31-0	COMP3	0-FFFF FFFFh	Compare 3. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. A read of this register will return the current compare value. A write to this register will provide a new compare value.

12.3.21 RTI Update Compare 3 Register (RTIUDCP3)

The update compare 3 register holds the value to be added to the compare register 3 value on a compare match. This register is shown in Figure 12-28 and described in Table 12-22.

Figure 12-28. RTI Update Compare 3 Register (RTIUDCP3) [offset = 6Ch]

31	16
UDCP3	
R/WP-0	

15	0
UDCP3	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

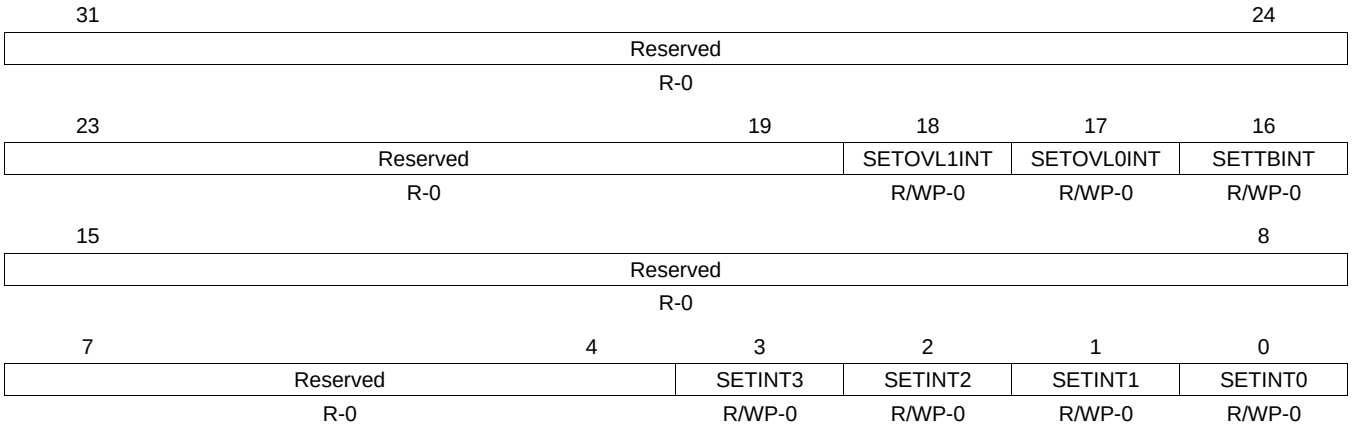
Table 12-22. RTI Update Compare 3 Register (RTIUDCP3) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP3	0-FFFF FFFFh	Update compare 3. This register holds a value that is added to the value in the RTICOMP3 register each time a compare matches. This process makes it possible to generate periodic interrupts without software intervention. A read of this register will return the value to be added to the RTICOMP3 register on the next compare match. A write to this register will provide a new update value.

12.3.22 RTI Set Interrupt Enable Register (RTISETINTENA)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be enabled. This register is shown in Figure 12-29 and described in Table 12-23.

Figure 12-29. RTI Set Interrupt Control Register (RTISETINTENA) [offset = 80h]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-23. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18	SETOVL1INT		Set free running counter 1 overflow interrupt.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read or Write: Interrupt is enabled.
17	SETOVL0INT		Set free running counter 0 overflow interrupt.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read or Write: Interrupt is enabled.
16	SETTBINT		Set timebase interrupt.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read or Write: Interrupt is enabled.
15-4	Reserved	0	Reads return 0. Writes have no effect.
3	SETINT3		Set compare interrupt 3.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read or Write: Interrupt is enabled.
2	SETINT2		Set compare interrupt 2.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read or Write: Interrupt is enabled.
1	SETINT1		Set compare interrupt 1.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read or Write: Interrupt is enabled.

Table 12-23. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions (continued)

Bit	Field	Value	Description
0	SETINT0		Set compare interrupt 0.
		0	Read: Interrupt is disabled. Write: Corresponding bit is unchanged.
		1	Read or Write: Interrupt is enabled.

12.3.23 RTI Clear Interrupt Enable Register (RTICLEARINTENA)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be disabled. This register is shown in Figure 12-30 and described in Table 12-24.

Figure 12-30. RTI Clear Interrupt Control Register (RTICLEARINTENA) [offset = 84h]

31	24
Reserved	
R-0	

23	19	18	17	16
Reserved		CLEAROVL1INT	CLEAROVL0INT	CLEARTBINT
R-0		R/WP-0	R/WP-0	R/WP-0

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		CLEARINT3	CLEARINT2	CLEARINT1	CLEARINT0
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-24. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18	CLEAROVL1INT		Clear free running counter 1 overflow interrupt.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read: Interrupt is enabled.
			Write: Interrupt is disabled.
17	CLEAROVL0INT		Clear free running counter 0 overflow interrupt.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read: Interrupt is enabled.
			Write: Interrupt is disabled.
16	CLEARTBINT		Clear timebase interrupt.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read: Interrupt is enabled.
			Write: Interrupt is disabled.
15-4	Reserved	0	Reads return 0. Writes have no effect.
3	CLEARINT3		Clear compare interrupt 3.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read: Interrupt is enabled.
			Write: Interrupt is disabled.
2	CLEARINT2		Clear compare interrupt 2.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read: Interrupt is enabled.
			Write: Interrupt is disabled.

Table 12-24. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions (continued)

Bit	Field	Value	Description
1	CLEARINT1		Clear compare interrupt 1.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read: Interrupt is enabled.
			Write: Interrupt is disabled.
0	CLEARINT0		Clear compare interrupt 0.
		0	Read: Interrupt is disabled.
			Write: Corresponding bit is unchanged.
		1	Read: Interrupt is enabled.
			Write: Interrupt is disabled.

12.3.24 RTI Interrupt Flag Register (RTIINTFLAG)

The corresponding flags are set at every compare match of the RTIFRCx and RTICOMPx values, whether the interrupt is enabled or not. This register is shown in Figure 12-31 and described in Table 12-25.

Figure 12-31. RTI Interrupt Flag Register (RTIINTFLAG) [offset = 88h]

31	24
Reserved	
R-0	

23	19	18	17	16
Reserved		OVL1INT	OVL0INT	TBINT
R-0		R/W1CP-0	R/W1CP-0	R/W1CP-0

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		INT3	INT2	INT1	INT0
R-0		R/W1CP-0	R/W1CP-0	R/W1CP-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 12-25. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18	OVL1INT		Free running counter 1 overflow interrupt flag. This bit determines if an interrupt is pending.
		0	Read: No interrupt is pending. Write: Bit is unchanged.
		1	Read: Interrupt is pending. Write: Bit is cleared to 0.
17	OVL0INT		Free running counter 0 overflow interrupt flag. This bit determines if an interrupt is pending.
		0	Read: No interrupt is pending. Write: Bit is unchanged.
		1	Read: Interrupt is pending. Write: Bit is cleared to 0.
16	TBINT		Timebase interrupt flag. This flag is set when the TBEXT bit is cleared by detection of a missing external clock edge. It will not be set by clearing TBEXT by software. It determines if an interrupt is pending.
		0	Read: No interrupt is pending. Write: Bit is unchanged.
		1	Read: Interrupt is pending. Write: Bit is cleared to 0.
15-4	Reserved	0	Reads return 0. Writes have no effect.
3	INT3		Interrupt flag 3. These bits determine if an interrupt due to a Compare 3 match is pending.
		0	Read: No interrupt is pending. Write: Bit is unchanged.
		1	Read: Interrupt is pending. Write: Bit is cleared to 0.
2	INT2		Interrupt flag 2. These bits determine if an interrupt due to a Compare 2 match is pending.
		0	Read: No interrupt is pending. Write: Bit is unchanged.
		1	Read: Interrupt is pending. Write: Bit is cleared to 0.

Table 12-25. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions (continued)

Bit	Field	Value	Description
1	INT1		Interrupt flag 1. These bits determine if an interrupt due to a Compare 1 match is pending.
		0	Read: No interrupt is pending. Write: Bit is unchanged.
		1	Read: Interrupt is pending. Write: Bit is cleared to 0.
0	INT0		Interrupt flag 0. These bits determine if an interrupt due to a Compare 0 match is pending.
		0	Read: No interrupt is pending. Write: Bit is unchanged.
		1	Read: Interrupt is pending. Write: Bit is cleared to 0.

12.3.25 Digital Watchdog Control Register (RTIDWDCTRL)

The software has to write to the DWDCTRL field in order to enable the DWD, as described below. Once enabled, the watchdog can only be disabled by a system reset. The application cannot disable the watchdog. However should the RTICLK source be changed to a source that is unimplemented it will have the same effect as disabling the watchdog. This register is shown in Figure 12-31 and described in Table 12-25.

Figure 12-32. Digital Watchdog Control Register (RTIDWDCTRL) [offset = 90h]

31	16
DWDCTRL	
R/WP-5312h	

15	0
DWDCTRL	
R/WP-ACEDh	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

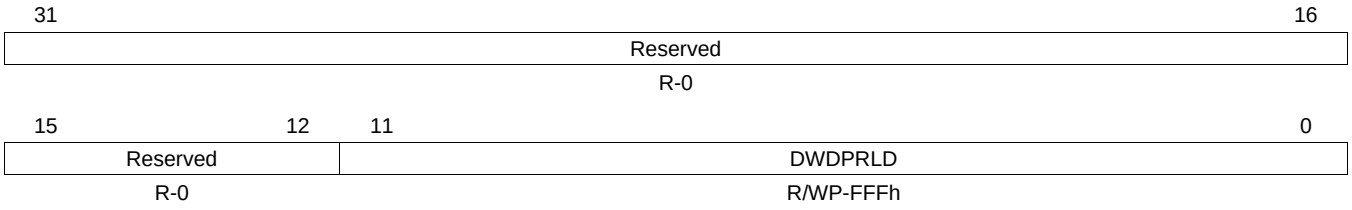
Table 12-26. Digital Watchdog Control Register (RTIDWDCTRL) Field Descriptions

Bit	Field	Value	Description
31-0	DWDCTRL		Digital Watchdog Control.
		5312 ACEDh	Read: DWD counter is disabled. Write: State of DWD counter is unchanged (stays enabled or disabled).
		A985 59DAh	Read: DWD counter is enabled. Write: DWD counter is enabled.
		All other values	Read: DWD counter state is unchanged (enabled or disabled). Write: State of DWD counter is unchanged (stays enabled or disabled). **Note:** Once the enable value is written, all other future writes are blocked. In other words, once DWD is enabled, it can only be disabled by system reset or power on reset. However should the RTICLK source be changed to a source that is unimplemented it will have the same effect as disabling the watchdog.

12.3.26 Digital Watchdog Preload Register (RTIDWDPRLD)

This register sets the expiration time of the DWD. This register is shown in Figure 12-31 and described in Table 12-25.

Figure 12-33. Digital Watchdog Preload Register (RTIDWDPRLD) [offset = 94h]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-27. Digital Watchdog Preload Register (RTIDWDPRLD) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0 and writes have no effect.
11-0	DWDPRLD	0-FFFh	Digital Watchdog Preload Value. Read: The current preload value. Write: Set the preload value. The DWD preload register can be configured only when the DWD is disabled. Therefore, the application can only configure the DWD preload register before it enables the DWD down counter. The expiration time of the DWD Down Counter can be determined with following equation: texp = (DWDPRLD+1) × 2^{13} / RTICLK1 where: DWDPRLD = 0...4095

12.3.27 Watchdog Status Register (RTIWDSTATUS)

This register records the status of the DWD. The values of the following status bits will not be affected by a soft reset. These bits are cleared by a power-on reset, or by a write of 1. These bits can be used for debug purposes. This register is shown in Figure 12-31 and described in Table 12-25.

Figure 12-34. Watchdog Status Register (RTIWDSTATUS) [offset = 98h]

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved		DWWD ST	END TIME VIOL	START TIME VIOL	KEY ST	DWD ST	Reserved
R-0		R/W1CP-x	R/W1CP-x	R/W1CP-x	R/W1CP-x	R/W1CP-x	R-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; x = value is unknown after reset; -*n* = value after reset

Table 12-28. Watchdog Status Register (RTIWDSTATUS) Field Descriptions

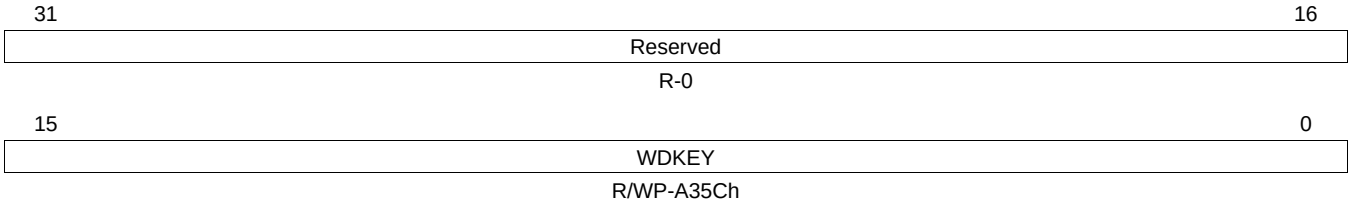
Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5	DWWD ST		Windowed Watchdog Status.
		0	Read: No time-window violation has occurred. Write: Leaves the current value unchanged.
		1	Read: Time-window violation has occurred. The watchdog has generated either a system reset or a non-maskable interrupt to the CPU in this case. Write: Bit is cleared to 0. This will also clear all other status flags in the RTIWDSTATUS register. Clearing of the status flags will deassert the non-maskable interrupt generated due to violation of the DWWD.
4	END TIME VIOL		Windowed Watchdog End Time Violation Status. This bit indicates whether the Watchdog counter expired.
		0	Read: No end-time window violation has occurred. Write: Leaves the current value unchanged.
		1	Read: End-time defined by the windowed watchdog configuration has been violated. Write: Bit is cleared to 0.
3	START TIME VIOL		Windowed Watchdog Start Time Violation Status. This bit indicates whether the key is written before the watchdog window opened up.
		0	Read: No start-time window violation has occurred. Write: Leaves the current value unchanged.
		1	Read: Start-time defined by the windowed watchdog configuration has been violated. Write: Bit is cleared to 0.
2	KEY ST		Watchdog key status. This bit indicates a reset or NMI generated by a wrong key or key sequence written to the RTIWDKEY register.
		0	Read: No wrong key or key-sequence written. Write: Bit is unchanged.
		1	Read: Wrong key or key-sequence written to RTIWDKEY register. Write: Bit is cleared to 0.
1	DWD ST		DWD status. This bit is equivalent to bit END TIME VIOL.
		0	Read: No reset or NMI was generated. Write: Bit is unchanged.
		1	Read: Reset or NMI was generated. Write: Bit is cleared to 0.
0	Reserved	0	Reads return 0. Writes have no effect.

12.3.28 RTI Watchdog Key Register (RTIWDKEY)

This register must be written with the correct written key values to serve the watchdog. This register is shown in Figure 12-35 and described in Table 12-29.

NOTE: It has to be taken into account that the write to the RTIWDKEY register takes 3 VCLK cycles.

Figure 12-35. RTI Watchdog Key Register (RTIDWDKEY) [offset = 9Ch]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-29. RTI Watchdog Key Register (RTIDWDKEY) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0 and writes have no effect.
15-0	WDKEY	0-FFFFh	Watchdog key. These bits provide the key sequence location. Reads returns the current WDKEY value. A write of E51Ah followed by A35Ch in two separate write operations defines the key sequence and reloads the DWD. Writing any other value causes a reset or NMI, as shown in Table 12-30. Writing any other value will cause the WDKEY to reset to A35Ch.

Table 12-30. Example of a WDKEY Sequence

Step	Value Written to WDKEY	Result
1	A35Ch	No action
2	A35Ch	No action
3	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
4	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
5	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
6	A35Ch	Watchdog is reset.
7	A35Ch	No action
8	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
9	A35Ch	Watchdog is reset.
10	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
11	2345h	System reset or NMI; incorrect value written to WDKEY.

12.3.29 RTI Digital Watchdog Down Counter (RTIDWDCNTR)

This register provides the current value of the DWD down counter. This register is shown in Figure 12-36 and described in Table 12-31.

Figure 12-36. RTI Watchdog Down Counter Register (RTIDWDCNTR) [offset = A0h]

31	25	24	16
Reserved		DWDCNTR	
R-0		R-1FFh	

15	0
DWDCNTR	
R-FFFFh	

LEGEND: R = Read only; -*n* = value after reset

Table 12-31. RTI Watchdog Down Counter Register (RTIDWDCNTR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0 and writes have no effect.
24-0	DWDCNTR	0-1FF FFFFh	DWD down counter. Reads return the current counter value.

12.3.30 Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL)

This register selects the DWWD reaction if the watchdog is serviced outside the time window. This register is shown in Figure 12-37 and described in Table 12-32.

Figure 12-37. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) [offset = A4h]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		WWDRXN	
R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-32. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0 and writes have no effect.
3-0	WWDRXN		The DWWD reaction.
		5h	The windowed watchdog will cause a reset if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
		Ah	The windowed watchdog will generate a non-maskable interrupt to the CPU if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
		All other values	The windowed watchdog will cause a reset if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
			Note: The DWWD reaction can be selected by the application even when the DWWD counter is already enabled. If a change to the WWDRXN is made before the watchdog service window is opened, then the change in the configuration takes effect immediately. If a change to the WWDRXN is made when the watchdog service window is already open, then the change in configuration takes effect only after the watchdog is serviced.

12.3.31 Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL)

This register selects the DWWD window size. This register is shown in Figure 12-38 and described in Table 12-33.

Figure 12-38. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) [offset = A8h]

31	16
WWDSIZE	
R-0000	

15	0
WWDSIZE	
R/WP-0005h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-33. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) Field Descriptions

Bit	Field	Value	Description
31-0	WWDSIZE	0	The DWWD window size.
		0000 0005h	100% (The functionality is the same as the standard time-out digital watchdog.)
		0000 0050h	50%
		0000 0500h	25%
		0000 5000h	12.5%
		0005 0000h	6.25%
		All other values	3.125%
			Note: The DWWD window size can be selected by the application even when the DWWD counter is already enabled. If a change to the WWDSIZE is made before the watchdog service window is opened, then the change in the configuration takes effect immediately. If a change to the WWDSIZE is made when the watchdog service window is already open, then the change in configuration takes effect only after the watchdog is serviced.

12.3.32 RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE)

This register enables an "auto-clear" of the compare interrupt enable bit after a compare equal event. This register is shown in Figure 12-39 and described in Table 12-34.

Figure 12-39. RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE) [offset = ACh]

31 28	27 24	23 20	19 16
Reserved	INTCLRENABLE3	Reserved	INTCLRENABLE2
R-0	R/WP-5h	R-0	R/WP-5h
15 12	**11 8**	**7 4**	**3 0**
Reserved	INTCLRENABLE1	Reserved	INTCLRENABLE0
R-0	R/WP-5h	R-0	R/WP-5h

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 12-34. RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE)

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	INTCLRENABLE3	5h	Enables the auto-clear functionality on the compare 3 interrupt. Read: Auto-clear for compare 3 interrupt is disabled. Privileged Write: Auto-clear for compare 3 interrupt becomes disabled.
		All other values	Read: Auto-clear for compare 3 interrupt is enabled. Privileged Write: Auto-clear for compare 3 interrupt becomes enabled.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	INTCLRENABLE2	5h	Enables the auto-clear functionality on the compare 2 interrupt. Read: Auto-clear for compare 2interrupt is disabled. Privileged Write: Auto-clear for compare 2 interrupt becomes disabled.
		All other values	Read: Auto-clear for compare 2 interrupt is enabled. Privileged Write: Auto-clear for compare 2 interrupt becomes enabled.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	INTCLRENABLE1	5h	Enables the auto-clear functionality on the compare 1 interrupt. Read: Auto-clear for compare 1 interrupt is disabled. Privileged Write: Auto-clear for compare 1 interrupt becomes disabled.
		All other values	Read: Auto-clear for compare 1 interrupt is enabled. Privileged Write: Auto-clear for compare 1 interrupt becomes enabled.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	INTCLRENABLE0	5h	Enables the auto-clear functionality on the compare 0 interrupt. Read: Auto-clear for compare 0 interrupt is disabled. Privileged Write: Auto-clear for compare 0 interrupt becomes disabled.
		All other values	Read: Auto-clear for compare 0 interrupt is enabled. Privileged Write: Auto-clear for compare 0 interrupt becomes enabled.

12.3.33 RTI Compare 0 Clear Register (RTICMP0CLR)

This registers holds a compare value, which is compared with the counter selected in the RTICOMPCTRL register Section 12.3.3. If the Free Running Counter matches the compare value, the compare 0 interrupt or DMA request line is cleared and the value in the RTIUDCP0 register Section 12.3.15 is added to this register. This register is shown in Figure 12-40 and described in Table 12-35.

Figure 12-40. RTI Compare 0 Clear Register (RTICMP0CLR) [offset = B0h]

31	16
CMP0CLR	
R/WP-0	

15	0
CMP0CLR	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-35. RTI Compare 0 Clear Register (RTICMP0CLR)

Bit	Field	Value	Description
31-0	CMP0CLR	0-FFFF FFFFh	Compare 0 clear. This registers holds a compare value, which is compared with the counter selected in the RTICOMPCTRL register. If the Free Running Counter matches the compare value, the compare 0 interrupt or DMA request line is cleared and the value in the RTIUDCP0 register Section 12.3.15 is added to this register. Reads return the current compare clear value. A privileged write to this register updates the compare clear value.

12.3.34 RTI Compare 1 Clear Register (RTICMP1CLR)

This registers holds a compare value, which is compared with the counter selected in the RTICOMPCTRL register Section 12.3.3. If the Free Running Counter matches the compare value, the compare 0 interrupt or DMA request line is cleared and the value in the RTIUDCP1 register Section 12.3.17 is added to this register. This register is shown in Figure 12-41 and described in Table 12-36.

Figure 12-41. RTI Compare 1 Clear Register (RTICMP1CLR) [offset = B4h]

31	16
CMP1CLR	
R/WP-0	

15	0
CMP1CLR	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-36. RTI Compare 1 Clear Register (RTICMP1CLR)

Bit	Field	Value	Description
31-0	CMP0CLR	0-FFFF FFFFh	Compare 1 clear. This registers holds a compare value, which is compared with the counter selected in the RTICOMPCTRL register. If the Free Running Counter matches the compare value, the compare 1 interrupt or DMA request line is cleared and the value in the RTIUDCP1 register Section 12.3.17 is added to this register. Reads return the current compare clear value. A privileged write to this register updates the compare clear value.

12.3.35 RTI Compare 2 Clear Register (RTICMP2CLR)

This registers holds a compare value, which is compared with the counter selected in the RTICOMPCTRL register Section 12.3.3. If the Free Running Counter matches the compare value, the compare 2 interrupt or DMA request line is cleared and the value in the RTIUDCP2 register Section 12.3.19 is added to this register. This register is shown in Figure 12-42 and described in Table 12-37.

Figure 12-42. RTI Compare 2 Clear Register (RTICMP2CLR) [offset = B8h]

31	16
CMP2CLR	
R/WP-0	

15	0
CMP2CLR	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-37. RTI Compare 2 Clear Register (RTICMP2CLR)

Bit	Field	Value	Description
31-0	CMP2CLR	0-FFFF FFFFh	Compare 2 clear. This registers holds a compare value, which is compared with the counter selected in the RTICOMPCTRL register. If the Free Running Counter matches the compare value, the compare 2 interrupt or DMA request line is cleared and the value in the RTIUDCP2 register Section 12.3.19 is added to this register. Reads return the current compare clear value. A privileged write to this register updates the compare clear value.

12.3.36 RTI Compare 3 Clear Register (RTICMP3CLR)

This registers holds a compare value, which is compared with the counter selected in the RTICOMPCTRL register Section 12.3.3. If the Free Running Counter matches the compare value, the compare 3 interrupt or DMA request line is cleared and the value in the RTIUDCP3 register Section 12.3.21 is added to this register. This register is shown in Figure 12-43 and described in Table 12-38.

Figure 12-43. RTI Compare 3 Clear Register (RTICMP3CLR) [offset = BCh]

31	16
CMP3CLR	
R/WP-0	

15	0
CMP3CLR	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 12-38. RTI Compare 3 Clear Register (RTICMP3CLR)

Bit	Field	Value	Description
31-0	CMP3CLR	0-FFFF FFFFh	Compare 3 clear. This registers holds a compare value, which is compared with the counter selected in the RTICOMPCTRL register. If the Free Running Counter matches the compare value, the compare 3 interrupt or DMA request line is cleared and the value in the RTIUDCP3 register Section 12.3.21 is added to this register. Reads return the current compare clear value. A privileged write to this register updates the compare clear value.

Cyclic Redundancy Check (CRC) Controller Module

This chapter describes the cyclic redundancy check (CRC) controller module.

Topic **Page**

13.1 Overview

The CRC controller is a module that is used to perform CRC (Cyclic Redundancy Check) to verify the integrity of a memory system. A signature representing the contents of the memory is obtained when the contents of the memory are read into the CRC controller. The responsibility of the CRC controller is to calculate the signature for a set of data and easy comparison of the calculated signature value against a pre-determined good signature value. The CRC controller supports two channels to perform CRC calculations on multiple memories in parallel and can be used on any memory system.

13.1.1 Features

The CRC controller offers:

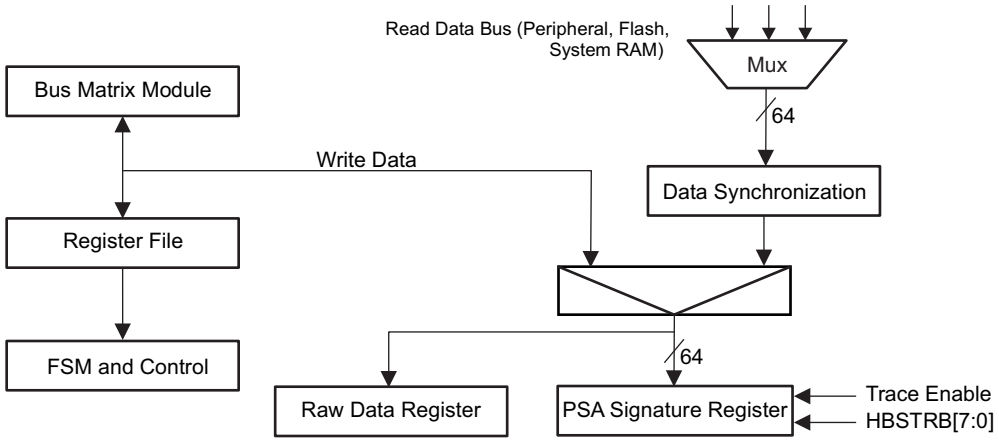
- Two channels to perform background signature verification on any memory sub-system.
- Data compression on 8-, 16-, 32-, and 64-bit data size.
- Maximum-length PSA (Parallel Signature Analysis) register constructed based on 64 bit primitive polynomial.
- Support for Full-CPU mode of operation.
- Data trace capability on Peripheral Bus Master, Flash, and System RAM data buses.

13.1.2 Block Diagram

Figure 13-1 shows a block diagram of the CRC controller.

NOTE: Only Channel 1 can support data trace. See Section 13.2.5.

Figure 13-1. CRC Controller Block Diagram For One Channel



13.2 Module Operation

13.2.1 General Operation

There are two channels in the CRC controller with each having a memory-mapped PSA (Parallel Signature Analysis) Signature Register. The CRC Controller can calculate the PSA signature over any memory configuration or memory space and data patterns defined by the application using the Full CPU mode of the CRC Controller. A data pattern can be 8-, 16-, 32-, or 64-bit data.

The CRC module performs the signature calculation which can then be compared with a pre-determined value to determine a pass or fail criteria. When a data pattern is written to the PSA Signature Register it is compressed into a signature. Once all of the chosen data has been written to the PSA Signature Register, a read of the PSA Signature Register will return the calculated signature for the data.

The CRC Controller also provides data trace capability using CRC Channel 1. During data trace, channel 1 monitors any data being read on the CPU data bus and compresses it and the results are shown in the PSA Signature Register.

13.2.2 CRC Modes of Operation

For this device, the CRC module supports Full CPU mode of operation with or without data trace enabled. In other TMS570 devices that include the DMA feature, Auto and Semi-Auto modes are supported in addition to Full CPU mode.

13.2.2.1 Full CPU Mode

In Full-CPU mode, the CPU is used to transfer the data patterns and perform the final signature verification. In this mode, the data patterns are read from memory and written into the PSA Signature Register. After the data patterns are compressed using the PSA Signature register, the CPU can read the calculated result from the PSA Signature Register and compare the calculated result to the pre-determined CRC signature value.

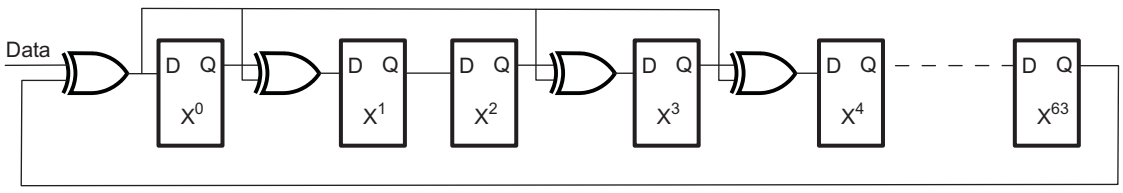
13.2.3 PSA Signature Register

The 64-bit PSA Signature Register is based on the primitive polynomial (as in the following equation) to produce the maximum length LFSR (Linear Feedback Shift Register), as shown in Figure 13-2.

$$f(x) = x^{64} + x^4 + x^3 + x + 1 \tag{25}$$

The serial implementation of LFSR has a limitation that, it requires *n* clock cycles to calculate the CRC values for an *n*-bit data stream. The idea is to produce the same CRC value operating on a multi-bit data stream, as would occur if the CRC were computed one bit at a time over the whole data stream. The algorithm involves looping to simulate the shifting, and concatenating strings to build the equations after *n* shift.

Figure 13-2. Linear Feedback Shift Register (LFSR)



The parallel CRC calculation based on the polynomial can be illustrated in the following HDL code:

```
for i in 63 to 0 loop
   NEXT_CRC_VAL(0) := CRC_VAL(63) xor DATA(i);
     for j in 1 to 63 loop
       case j is
          when 1|3|4 =>
               NEXT_CRC_VAL(j) :=
                   CRC_VAL(j - 1) xor CRC_VAL(63) xor DATA(i);
          when others =>
               NEXT_CRC_VAL(j) := CRC_VAL(j - 1);
       end case;
     end loop;
   CRC_VAL := NEXT_CRC_VAL;
end loop;
```

NOTE: 1) The inner loop is to calculate the next value of each shift register bit after one cycle

2) The outer loop is to simulate 64 cycles of shifting. The equation for each shift register bit is thus built before it is compressed into the shift register.

3) MSB of the DATA is shifted in first

There is one PSA Signature Register per CRC channel. The PSA Signature Register can be both read and written. When it is written, it can either compress the data or just capture the data depending on the state of CHx_MODE bits. If CHx_MODE=Data Capture, a seed value can be planted in the PSA Signature Register without compression. If not in the Data Capture mode, the data written will be compressed by the PSA Signature Register when it is written. Each channel can be seeded with different values before compression starts. When the PSA Signature Register is read, it gives the calculated signature.

In Full-CPU mode, the number of data patterns to be compressed is determined by the CPU/application and signature calculation will continue as long as data continues to be written to the PSA Signature Register or until a PSA Software Reset is generated for the channel using the CRC_CTRL0 register. When the the software reset is instantiated, the PSA Signature Register is reset to all zeros.

The CRC Controller supports double word, word, half word and byte access to the PSA Signature Register. During a non-double word write access, all unwritten bytes are padded with zero’s before compression. Note that comparison between PSA Sector Signature Register and CRC Value Register should always be completed using 64 bit accesses because a compressed value is always expressed as a 64 bit value.

The PSA Signature Register is reset to zero under the following conditions:

- System reset
- PSA Software reset
- One sector of data patterns are compressed

13.2.4 Raw Data Register

The raw or un-compressed data written to the PSA Signature Register is also saved in the Raw Data Register. This register is read only.

13.2.5 CPU Data Trace

CRC channel 1 can be used to trace (or "snoop") Flash, System RAM and Peripheral Bus Master data buses. However, at any one point, only one bus is traced. It is possible to disable the tracing of any of the buses by setting the appropriate bits in the CRC_TRACE_BUS_SEL register or all data tracing by writing zero to the CH1_TRACEEN bit of the CRC_CTRL2 register.

While tracing the data, there is a priority scheme implemented between the buses. The Peripheral Bus Master has the highest priority followed by Flash, and finally the System RAM. For each data read by the CPU on its data bus, the same data is compressed in the PSA Signature Register. In data trace mode, a write to the PSA Signature Register does not get compressed. Therefore, it is possible to write a seed value into PSA Signature Register before the bus tracing takes place. For non- double word reads on the data bus, all un-selected byte lanes are padded with zeros during compression.

13.2.5.1 Data Capture Mode used in Conjunction with Data Trace

Data capture mode is especially useful when it is used in conjunction when data trace for channel 1 is enabled (CRC_CTRL2.CH1_TRACEEN = 1). The seed value can be planted in PSA Signature Register during data capture mode by writing a seed value into PSA Signature Register. The data trace enable bit is then set to snoop and compress any read transaction on DAHB bus. When trace enable bit is set, CRC_CTRL2.CH1_MODE is automatically reset to 0 (data capture mode). To change from one mode to another mode in the middle of an on-going mode, perform the following steps:

1. Assert software reset for the respective channel.
2. Change from the current active mode to Data Capture mode (CRC_CTRL2.CH1_MODE = 0).
3. Change from Data Capture mode to the new mode.
4. Release software reset.

13.2.6 Power Down Mode

CRC module can be put into power down mode when the power down control bit PWDN is set. The module wakes up when the PWDN bit is cleared. When CRC controller is in power down mode, data trace is suspended. However, if CRC registers are accessed then data trace will resume.

13.2.7 Emulation

During emulation, registers and or memory spaces are often accessed in order to keep the information in the IDE up to date or refreshed. As a result, during emulation/debug, these accesses will not impact the functional operation of the module. As an example, if data trace mode is active, accesses by the CPU to data are not compressed by the PSA Signature Register when the device is in suspend mode.

13.2.8 Peripheral Bus Interface

The CRC Controller is a peripheral slave module and has a similar bus interface to all other modules on the device. The CRC Controller supports the following features:

- Different sizes of burst operation.
- Aligned and unaligned accesses.
- Abort is generated for any illegal address accesses.

13.3 Example/Hints: Full-CPU Mode

In a system without the availability of a DMA controller, the CRC routine needs to be controlled by the CPU and application software. The CPU needs to read from the memory area from which CRC is to be performed and, if not using data trace mode, the data will need to be written to the PSA Signature Register.

In a scenario where the memory space is very large or where bandwidth is a concern, the memory can be partitioned into blocks each with a pre-determined signature. When partitioned, the CPU can transfer the data to the PSA Signature Register on a cyclic basis so CPU resources are not overburdened by the CRC calculation.

In addition, efficiency can be gained by utilizing the data trace mode and performing 64-bit reads of the data over which the signature is to be calculated. When using this mode in a cyclic mode, data trace can be disabled after each block of data is checked and the signature calculation continued on the next cycle by re-enabling the data trace mode. Note, however, if data trace mode is used, care must be taken to insure interrupt servicing does not corrupt the signature calculation given CPU data accesses will most likely occur during interrupt servicing.

13.3.1 CRC Setup

For normal Full CPU mode of operation (without data trace enabled), the application software will need to configure the control registers for CRC operation. This is accomplished by first asserting the software reset using the bits for the PSA channel(s) to be utilized in the calculation (CRC_CTRL0 register) and setting the mode for Full CPU Mode using the CHx_MODE bits (CRC_CTRL2). If a seed value is desired, write the seed value to the PSA Signature Register before changing the Chx_MODE bits to Full CPU Mode as the CHx_MODE bits will default to the Data Capture mode setting which allows data to be written to the PSA Signature Register without compression.

Once the mode is established, read the first memory location/data pattern to be included in the PSA signature calculation and write it to the PSA Signature Register and continue this cycle of reading the data patterns and writing to the PSA Signature Register until all of the data to be included in the signature are read and written. Once all data has been read an written, read the PSA Signature Register to obtain the final calculated signature and compare it with the expected signature to determine pass or fail.

Alternatively, data trace can be enabled by also setting the CH1_TRACEEN bit in the CRC_CTRL2 register and selecting the bus for the memory space from which the data will be read in the CRC_TRACE_BUS_SEL register. Once this is completed, the data can simply be read by the CPU without the need to write it to the PSA Signature Register. Once all of the data has been read, disable the data trace mode and obtain the final calculated signature from the PSA Signature Register and compare with the expected results for pass or fail determination.

13.4 CRC Control Registers

Table 13-1 lists the registers of the CRC Controller. All registers are in word boundary. 64-, 32-, 16-, and 8-bit write accesses are supported to all registers. The base address for the control registers is FE00 0000h.

Table 13-1. CRC Control Registers

Offset	Acronym	Register Description	Section
0h	CRC_CTRL0	CRC Global Control Register	Section 13.4.1
8h	CRC_CTRL1	CRC Global Control Register 1	Section 13.4.2
10h	CRC_CTRL2	CRC Global Control Register 2	Section 13.4.3
60h	PSA_SIGREGL1	Channel 1 PSA Signature Low Register	Section 13.4.4
64h	PSA_SIGREGH1	Channel 1 PSA Signature High Register	Section 13.4.5
78h	RAW_DATAREGL1	Channel 1 Raw Data Low Register	Section 13.4.6
7Ch	RAW_DATAREGH1	Channel 1 Raw Data High Register	Section 13.4.7
A0h	PSA_SIGREGL2	Channel 2 PSA Signature Low Register	Section 13.4.8
A4h	PSA_SIGREGH2	Channel 2 PSA Signature High Register	Section 13.4.9
B8h	RAW_DATAREGL2	Channel 2 Raw Data Low Register	Section 13.4.10
BCh	RAW_DATAREGH2	Channel 2 Raw Data High Register	Section 13.4.11
140h	CRC_TRACE_BUS_SEL	Data Bus Selection Register	Section 13.4.12

13.4.1 CRC Global Control Register 0 (CRC_CTRL0)

Figure 13-3. CRC Global Control Register 0 (CRC_CTRL0) [offset = 00h]

31	16
Reserved	
R-0	

15	9	8
Reserved		CH2_PSA_SWREST
R-0		R/W-0

7	1	0
Reserved		CH1_PSA_SWREST
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-2. CRC Global Control Register 0 (CRC_CTRL0) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	CH2_PSA_SWREST		Channel 2 PSA Software Reset. When set, the PSA Signature Register is reset to all 0. Software reset does not reset software reset bit itself. Therefore, CPU is required to clear this bit by writing a 0.
		0	PSA Signature Register is not reset.
		1	PSA Signature Register is reset.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	CH1_PSA_SWREST		Channel 1 PSA Software Reset. When set, the PSA Signature Register is reset to all 0. Software reset does not reset software reset bit itself. Therefore, CPU is required to clear this bit by writing a 0.
		0	PSA Signature Register is not reset.
		1	PSA Signature Register is reset.

13.4.2 CRC Global Control Register (CRC_CTRL1)

Figure 13-4. CRC Global Control Register 1 (CRC_CTRL1) [offset = 08h]

31	16
Reserved	
R-0	

15	1	0
Reserved		PWDN
R-0		RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-3. CRC Global Control Register 1 (CRC_CTRL1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	PWDN		Power Down. When set, CRC module is put in power-down mode.
		0	CRC is not in power-down mode.
		1	CRC is in power-down mode.

13.4.3 CRC Global Control Register 2 (CRC_CTRL2)

Figure 13-5. CRC Global Control Register 2 (CRC_CTRL2) [offset = 10h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved						CH2_MODE	
R-0						R/WP-0	

7	6	5	4	3	2	1	0
Reserved			CH1_TRACEEN	Reserved		CH1_MODE	
R-0			R/WP-0	R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 13-4. CRC Global Control Register 2 (CRC_CTRL2) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	CH2_MODE		Channel 2 Mode Selection.
		0	Data Capture mode.
		1h-2h	Reserved
		3h	Full CPU mode
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	CH1_TRACEEN		Channel 1 Data Trace Enable. When set, the channel is put into data trace mode. The channel traces (or "snoops") on the CPU Peripheral Bus Master, Flash, or System RAM buses (selected through the CRC_TRACE_BUS_SEL register) for any read transaction. Any read data on these buses is compressed by the PSA Signature Register. When suspend is on, the PSA Signature Register does not compress any read data on these buses. When trace enable bit is set, the CH1_MODE bit is automatically reset to 0 (Data Capture mode).
		0	Data Trace is disabled.
		1	Data Trace is enabled.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1-0	CH1_MODE		Channel 1 Mode Selection.
		0	Data Capture mode.
		1h-2h	Reserved
		3h	Full CPU mode

13.4.4 Channel 1 PSA Signature Low Register (PSA_SIGREGL1)

Figure 13-6. Channel 1 PSA Signature Low Register (PSA_SIGREGL1) [offset = 60h]

31	0
PSASIG1	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 13-5. Channel 1 PSA Signature Low Register (PSA_SIGREGL1) Field Descriptions

Bit	Field	Description
31-0	PSASIG1	Channel 1 PSA Signature Low Register. This register contains the value stored at PSASIG1[31:0] register.

13.4.5 Channel 1 PSA Signature High Register (PSA_SIGREGH1)

Figure 13-7. Channel 1 PSA Signature High Register (PSA_SIGREGH1) [offset = 64h]

31	0
PSASIG1	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 13-6. Channel 1 PSA Signature High Register (PSA_SIGREGH1) Field Descriptions

Bit	Field	Description
31-0	PSASIG1	Channel 1 PSA Signature High Register. This register contains the value stored at PSASIG1[63:32] register.

13.4.6 Channel 1 Raw Data Low Register (RAW_DATAREGL1)

Figure 13-8. Channel 1 Raw Data Low Register (RAW_DATAREGL1) [offset = 78h]

31	0
RAW_DATA1	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 13-7. Channel 1 Raw Data Low Register (RAW_DATAREGL1) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA1	Channel 1 Raw Data Low Register. This register contains bits 31:0 of the uncompressed raw data.

13.4.7 Channel 1 Raw Data High Register (RAW_DATAREGH1)

Figure 13-9. Channel 1 Raw Data High Register (RAW_DATAREGH1) [offset = 7Ch]

31	0
RAW_DATA1	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 13-8. Channel 1 Raw Data High Register (RAW_DATAREGH1) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA1	Channel 1 Raw Data High Register. This register contains bits 63:32 of the uncompressed raw data.

13.4.8 Channel 2 PSA Signature Low Register (PSA_SIGREGL2)

Figure 13-10. Channel 2 PSA Signature Low Register (PSA_SIGREGL2) [offset = A0h]

31	0
PSASIG2	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 13-9. Channel 2 PSA Signature Low Register (PSA_SIGREGL2) Field Descriptions

Bit	Field	Description
31-0	PSASIG2	Channel 2 PSA Signature Low Register. This register contains the value stored at PSASIG2[31:0] register.

13.4.9 Channel 2 PSA Signature High Register (PSA_SIGREGH2)

Figure 13-11. Channel 2 PSA Signature High Register (PSA_SIGREGH2) [offset = A4h]

31	0
PSASIG2	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 13-10. Channel 2 PSA Signature High Register (PSA_SIGREGH2) Field Descriptions

Bit	Field	Description
31-0	PSASIG2	Channel 2 PSA Signature High Register. This register contains the value stored at PSASIG2[63:32] register.

13.4.10 Channel 2 Raw Data Low Register (RAW_DATAREGL2)

Figure 13-12. Channel 2 Raw Data Low Register (RAW_DATAREGL2) [offset = B8h]

31	0
RAW_DATA2	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 13-11. Channel 2 Raw Data Low Register (RAW_DATAREGL2) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA2	Channel 2 Raw Data Low Register. This register contains bits 31:0 of the uncompressed raw data..

13.4.11 Channel 2 Raw Data High Register (RAW_DATAREGH2)

Figure 13-13. Channel 2 Raw Data High Register (RAW_DATAREGH2) [offset = BCh]

31	0
RAW_DATA2	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 13-12. Channel 2 Raw Data High Register (RAW_DATAREGH2) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA2	Channel 2 Raw Data High Register. This register contains bits 63:32 of the uncompressed raw data..

13.4.12 Data Bus Selection Register (CRC_TRACE_BUS_SEL)

Figure 13-14. Data Bus Selection Register (CRC_TRACE_BUS_SEL) [offset = 140h]

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		MEn	DTCMEn	ITCMEn
R-0		R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13-13. Data Bus Selection Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	MEn		Enables the tracing of Peripheral Bus Master.
		0	Tracing of Peripheral Bus Master has been disabled.
		1	Tracing of Peripheral Bus Master has been enabled.
1	DTCMEn		Enables the tracing of data TCM.
		0	Tracing of System Odd and Even RAM buses have been disabled.
		1	Tracing of System Odd and Even RAM buses have been enabled.
0	ITCMEn		Enables the tracing of instruction TCM.
		0	Tracing of Flash data bus has been disabled.
		1	Tracing of Flash data bus has been enabled.

Vectored Interrupt Manager (VIM) Module

This chapter describes the behavior of the vectored interrupt manager (VIM) module of the device family.

Topic **Page**

14.1 Overview

The vectored interrupt manager (VIM) provides hardware assistance for prioritizing and controlling the many interrupt sources present on a device. Interrupts are caused by events outside of the normal flow of program execution. Normally, these events require a timely response from the central processing unit (CPU); therefore, when an interrupt occurs, the CPU switches execution from the normal program flow to an interrupt service routine (ISR).

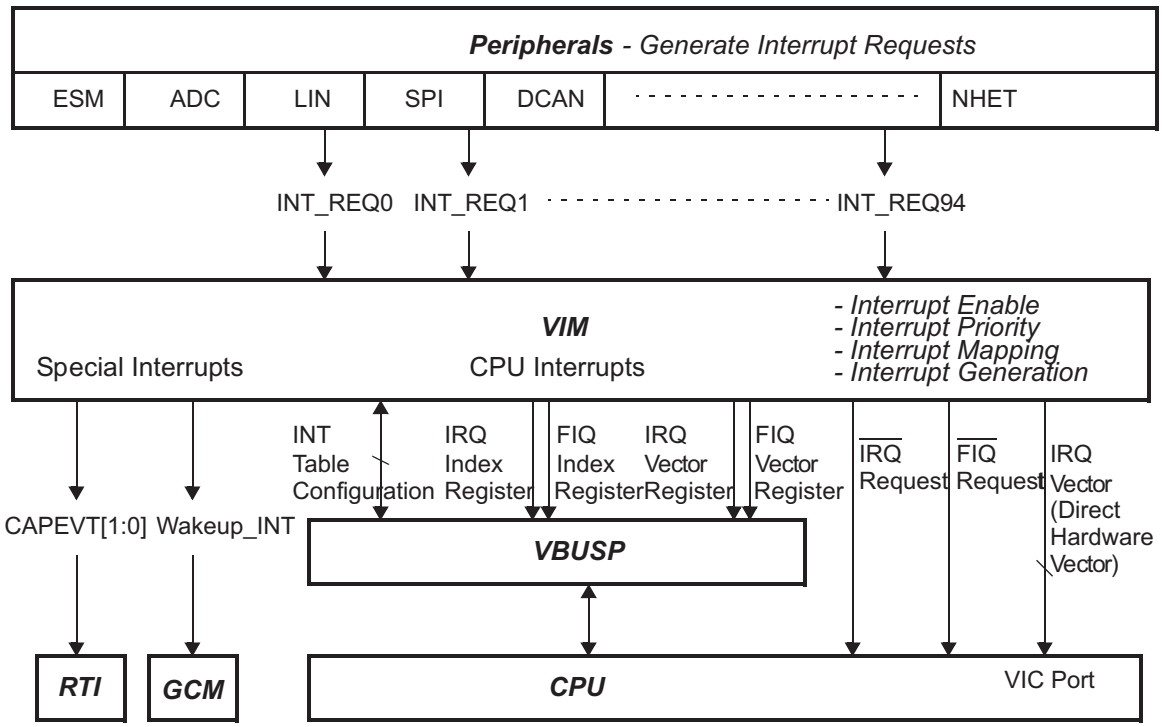
The VIM module has the following features:

- Supports 95 interrupt channels, in both register vectored interrupt and hardware vectored interrupt mode.
  - Provides IRQ vector directly to the CPU VIC port
  - Provides FIQ/IRQ vector through registers
  - Provides programmable priority and enable for interrupt request lines
- Provides a direct hardware dispatch mechanism for fastest IRQ dispatch.
- Provides two software dispatch mechanisms for backward compatibility with earlier generation of TI processors.
  - Index interrupt
  - Register vectored interrupt
- Parity protected vector interrupt table against soft errors.

14.2 Device Level Interrupt Management

A block diagram of device level interrupt handling is shown in Figure 14-1. When an event occurs within a peripheral, the peripheral makes an interrupt request to the VIM. Then, VIM prioritizes the requests from peripherals and provides the address of the highest interrupt service routine (ISR) to the CPU. Finally, CPU starts executing the ISR instructions from that address in the ISR. Section 14.2.1 through Section 14.2.3 provide additional details about these three steps.

Figure 14-1. Device Level Interrupt Block Diagram



14.2.1 Interrupt Generation at the Peripheral

Interrupt generation begins when an event occurs within a peripheral module. Some examples of interrupt-capable events are expiration of a counter within a timer module, receipt of a character in a communications module, and completion of a conversion in an analog-to-digital converter (ADC) module. Some device peripherals are capable of requesting interrupts on more than one interrupt request line.

Interrupts are not always generated when an event occurs; the peripheral must make an interrupt request to the VIM based on the event occurrence. Typically, the peripheral contains:

- An interrupt flag bit for each event to signify the event occurrence.
- An interrupt enable bit to control whether the event occurrence causes an interrupt request to the VIM.

14.2.2 Interrupt Handling at the CPU

The ARM CPU provides two vectors for interrupt requests—fast interrupt requests (FIQs) and normal interrupt requests (IRQs). FIQs are higher priority than IRQs, and FIQ interrupts may interrupt IRQ interrupts.

NOTE: The FIQ implemented in Cortex-R4F is Non-Maskable Fast Interrupts (NMFI). Once FIQ is enabled (by clearing F bit in CPSR), it can NOT be disabled by setting F bit in CPSR. Only a reset or an FIQ will be able to set the F bit in CPSR. By hardware, Non Maskable FIQ are not reentrant.

After reset (power reset or warm reset), both FIQ and IRQ are disabled. The CPU may enable these interrupt request channels individually within the CPSR (Current Program Status Register); CPSR bits 6 and 7 must be cleared to enable the FIQ (bit 6) and IRQ (bit 7) interrupt requests at the CPU. CPSR is writable in privilege mode only. Example 14-2 shows how to enable the IRQ and FIQ through CPSR.

When the CPU receives an interrupt request, the CPSR mode field changes to either FIQ or IRQ mode. When an IRQ interrupt is received, the CPU disables other IRQ interrupts by setting CPSR bit 7. When an FIQ interrupt is received, the CPU disables both IRQ and FIQ interrupts by setting CPSR bits 6 and 7.

A write of 1 to CPSR bit 7 disables the IRQ from CPU. However, a write of 1 to CPSR bit 6 leaves it unchanged. Example 14-2 also shows how to disable the IRQ through CPSR.

14.2.3 Software Interrupt Handling Options

The device supports three different possibilities for software to handle interrupts

1. Index interrupts mode (compatible with TMS470R1x legacy code)

   After the interrupt is received by the CPU, the CPU branches to 0x18 (IRQ) or 0x1C (FIQ) to execute the main ISR. The main ISR routine reads the offset register (IRQINDEX, FIQINDEX) to determine the source of the interrupt.

   This mode is compatible with the TMS470R1x (CIM) module and provides the same interrupt registers.

   This mode could be used if legacy code needs to be reused, porting it from the TMS470R1x family. However, imported software will not benefit from the VIM improvements.

   To port legacy software, the interrupt vector at 0x18 (IRQ) or 0x1C (FIQ) only needs to be a branch statement to a software interrupt table. The software interrupt table reads the pending interrupt from a vector offset register (FIQINDEX[7:0] for FIQ interrupts and IRQINDEX[7:0] for IRQ interrupts). All pending interrupts can be viewed in the INTREQ register. Example 14-4 shows how to respond to FIQ with short latency in this mode.

2. Register vectored interrupts (automatically provide vector address to application)

   Before enabling interrupts, the application software also has to initiate the interrupt vector table (VIM RAM).

   Once the VIM receives an interrupt, it loads the address of ISR from interrupt vector table, and store it into the interrupt vector register (IRQVECREG for IRQ interrupt, FIQVECREG for FIQ interrupt).

   After the interrupt is received by the CPU, the CPU executes the instruction placed at 0x18 or 0x1C (IRQ or FIQ vector) to load the address of ISR (interrupt vector) from the interrupt vector register. Example 14-3 illustrates the configuration for the exception vectors using this mode.

3. Hardware vectored interrupts (automatically dispatch to ISR, IRQ only)

   Before enabling interrupts, the application software must initiate the interrupt vector table (VIM RAM) pointing to the ISR for each interrupt channel.

   After the interrupt (IRQ) is received by the CPU, CPU reads the address of ISR directly from the interface with VIM (VIC port) instead of branching to 0x18. The CPU will branch directly to the ISR.

   The hardware vectored interrupt behavior must be explicitly enabled by setting the vector enable (VE) bit in the CP15 R1 register. This bit resets to 0, so that the default state after reset is backward compatible to earlier ARM CPU. Example 14-1 shows how to enable the hardware vectored interrupt.

   NOTE: This mode is NOT available for FIQ.

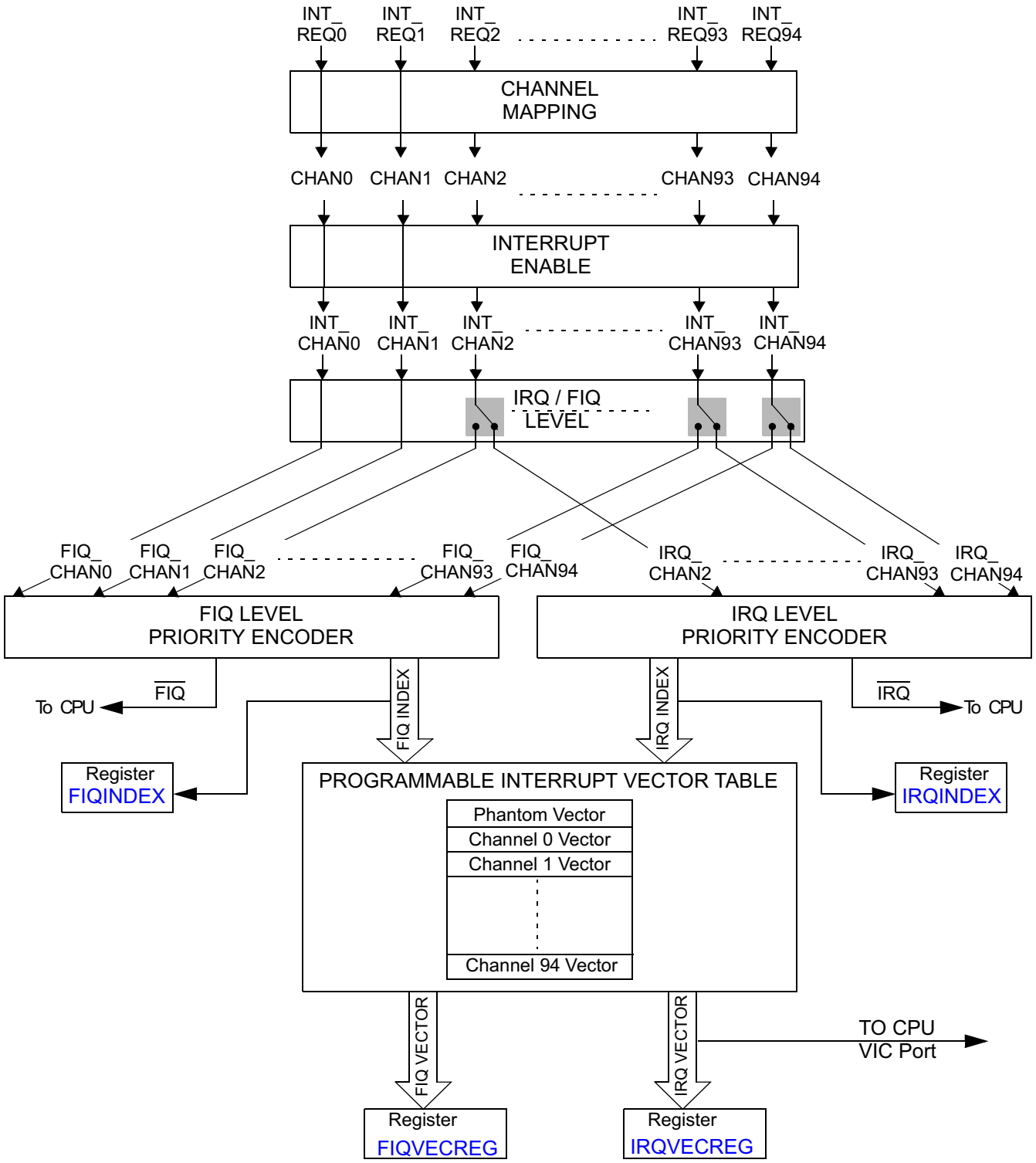
4. Software-Based Priority Decoding Scheme

   If the application uses a software-based interrupt priority decoding scheme instead of the hardware vector capabilities, then there is an additional step which was not required on earlier devices. This version of the VIM will hold an interrupt request generated by a peripheral. When the software clears the interrupt condition in the source module (for example, RTI, GIO, and so on), then it must also perform an additional clear of the interrupt request in the VIM. This can be done by reading the IRQVECREG register (Section 14.8.14) or FIQVECREG register (Section 14.8.15), or by writing a 1 to the INTREQ(i) bit (Section 14.8.9) in the VIM. This is not necessary if any of the three previous methods are used as the interrupt request bit in the VIM will be automatically cleared when the vector is read.

14.3 Interrupt Handling Inside VIM

A block diagram of the interrupt handling inside VIM is shown in Figure 14-2

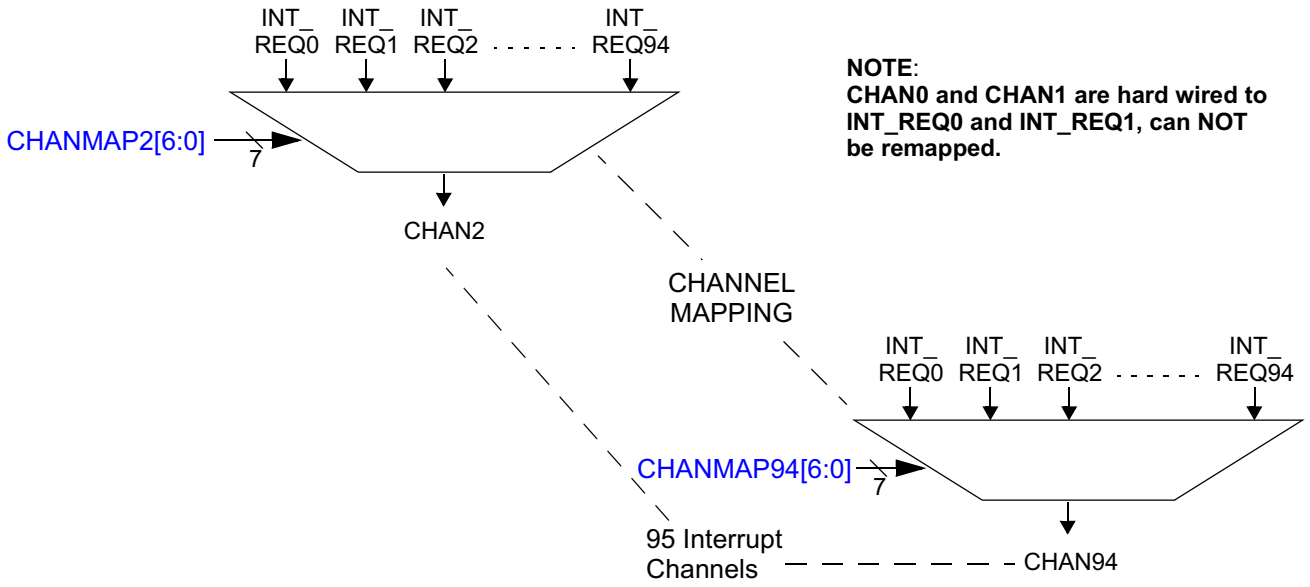
Figure 14-2. VIM Interrupt Handling Block Diagram



14.3.1 VIM Interrupt Channel Mapping

The VIM support 96 interrupt channels (including phantom interrupt). A block diagram of the VIM interrupt requests arrangement from peripheral modules to the interrupt channels, is provided in Figure 14-3. Each interrupt channel(CHANx) has a corresponding mapping register bit field (CHANMAPx[6:0]). This mapping register determines which interrupt channel it maps each VIM interrupt request. With this scheme, the same request can be mapped to multiple channels. A lower numbered channel in each FIQ and IRQ has higher priority. The programmability of the VIM allows software to control the interrupt priority.

Figure 14-3. VIM Channel Mapping



NOTE: CHAN95

CHAN95 has no dedicated interrupt vector table entry. Therefore, CHAN95 shall NOT be remapped to other INT_REQ (INT_REQ95 is reserved at device level).

In the reset state, the VIM maps all of the interrupt requests in the system to their respective interrupt channels. Figure 14-4 shows the default state following the reset.

Figure 14-5 shows the VIM INT2 is remapped to both Channel 2 and 4, and INT3 is mapped to channel 3.

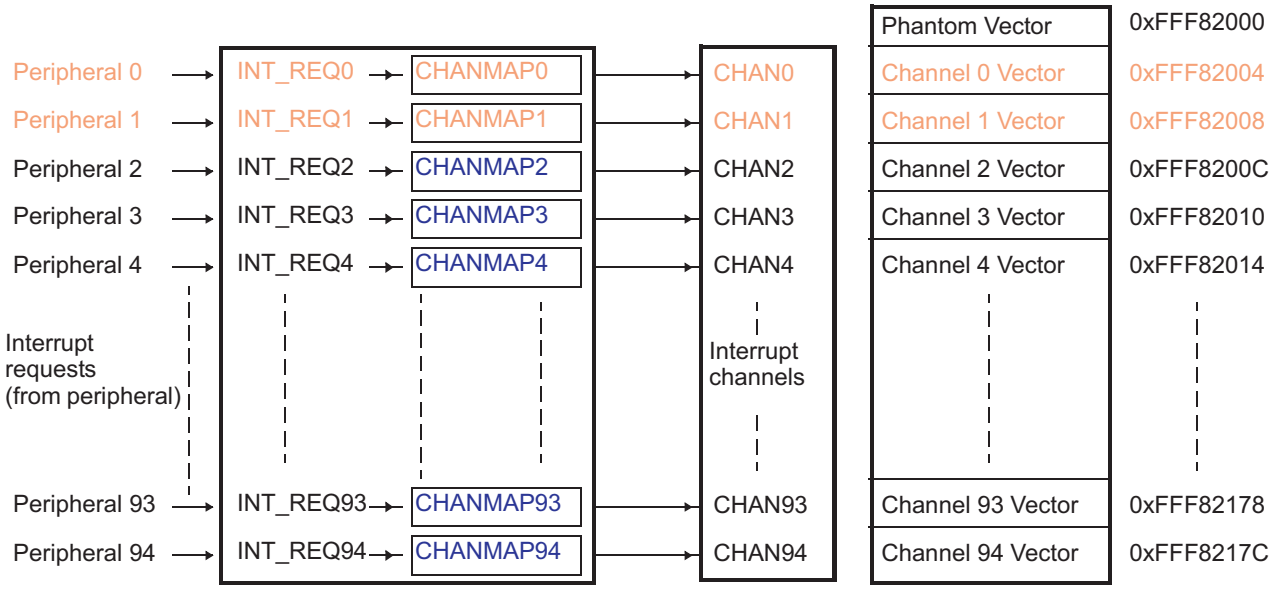
NOTE: By mapping INT2 to channel 2 and channel 4, and mapping INT3 to channel 3, it is possible for the software to change the priority dynamically by changing the ENABLE register (REQENASET and REQENACLR). When channel 2 is enabled, the priority is:

1. INT0
2. INT1
3. INT2
4. INT3

Disabling channel 2, the priority becomes:

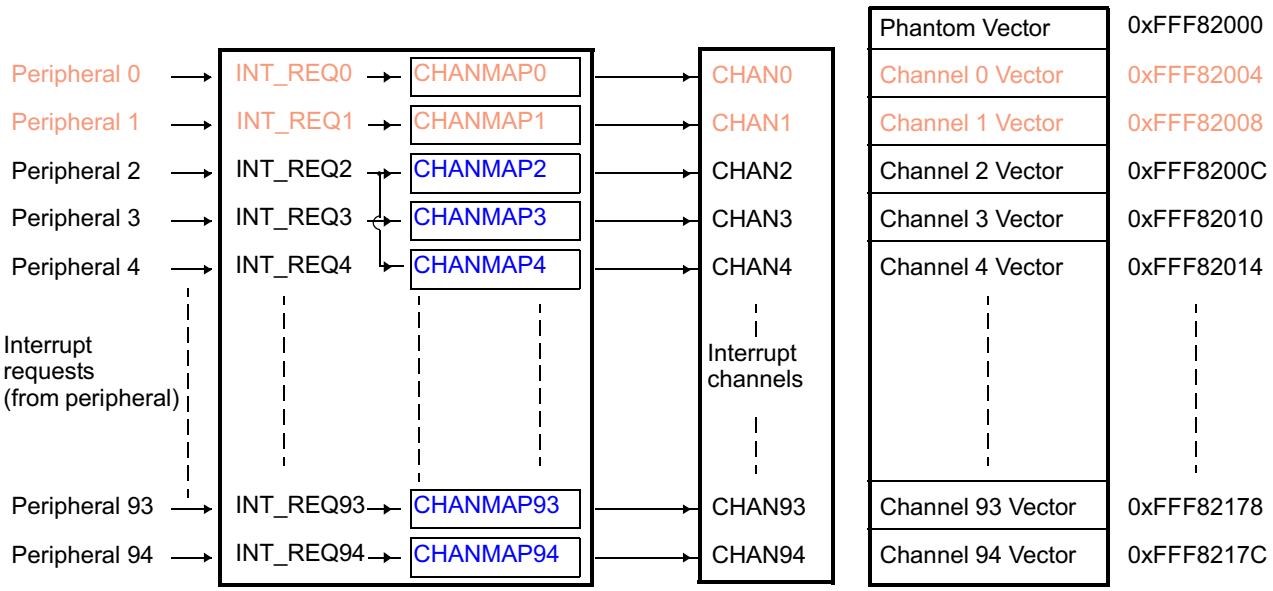
1. INT0
2. INT1
3. INT3
4. INT2

Figure 14-4. VIM in Default State



NOTE: CHAN0 and CHAN1 are hardwired to INT_REQ0 and INT_REQ1, so they cannot be remapped.

Figure 14-5. VIM in a Programmed State



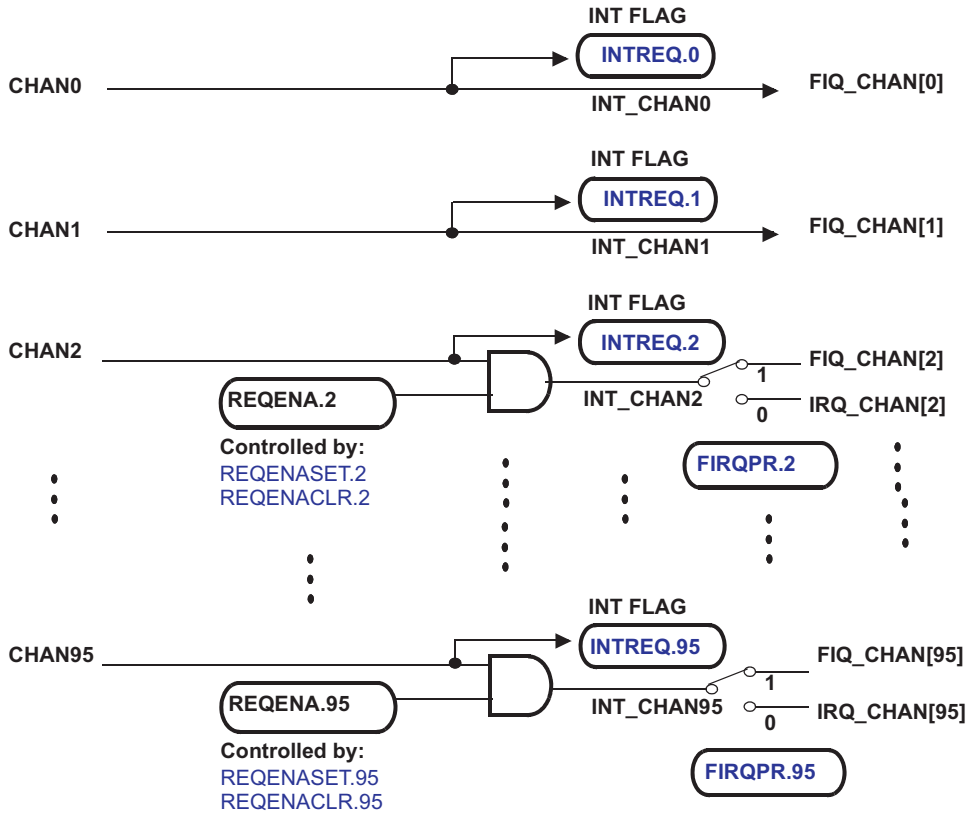
NOTE: CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1, so they cannot be remapped.

14.3.2 VIM Input Channel Management

As shown in Figure 14-6, the VIM enables channels on a channel-by-channel basis (in the REQENASET and REQENACLR registers); unused channels may be masked to prevent spurious interrupts.

NOTE: The interrupt ENABLE register does not affect the value of INTREQ.

Figure 14-6. Interrupt Channel Management



By default, interrupt CHAN0 is mapped to ESM (Error Signal Module) high level interrupt and CHAN1 is reserved for other NMI. For safety reasons, these two channels are mapped to FIQ only and can **NOT** be disabled through ENABLE registers.

NOTE: NMI Channel

Channel 0 and channel 1 are not maskable by the REQENASET/REQENACLR bit and both channels are routed exclusively to FIQ/NMI request line (FIRQPR0 and FIRQPR1 have no effect).

The VIM prioritizes the received interrupts based upon a programmed prioritization scheme. The VIM can send two interrupt requests to the CPU simultaneously—one IRQ and one FIQ. If both interrupt types are enabled at the CPU level, then the FIQ has greater priority and is handled first. Each interrupt channel, except channel 0 and 1, can be assigned to send either an FIQ or IRQ request to the CPU (in the FIRQPR register).

The VIM provides a default prioritization scheme, which sends the lowest numbered active channel (in each FIQ and IRQ classes) to the CPU. Within the FIQ and IRQ classes of interrupts, the lowest channel has the highest priority interrupt. The channel number is programmable through register CHANMAPx.

After the VIM has generated the vector corresponding to the highest active IRQ, it updates the FIQINDEX or the IRQINDEX register, depending on the class of interrupt. Then, it accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR. If the request is an FIQ class interrupt, the address read from the interrupt vector table, is written to the FIQVECREG register. If the request is an IRQ class interrupt, the address is written to the IRQVECREG register and put on the VIC port of the CPU (in case of hardware vectored interrupt is enabled).

All of the interrupt registers are updated when a new high priority interrupt line becomes active.

14.4 Interrupt Vector Table (VIM RAM)

Interrupt vector table stores the address of ISRs. During register vectored interrupt and hardware vectored interrupt, VIM accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR.

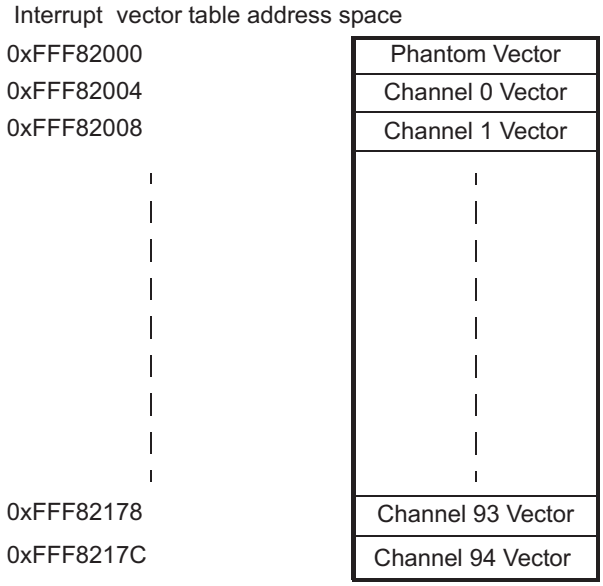
For safety reasons, the interrupt vector table has protection by parity to indicate corruption due to soft errors. The parity scheme is implemented as a continuous background check based on memory access. Section 14.4.1 through Section 14.4.4 describe how parity works in the interrupt vector table.

NOTE: Writes to the interrupt vector table parity flag register (PARFLG) and the interrupt vector table parity control register (PARCTL) are in privilege mode only.

14.4.1 Interrupt Vector Table Operation

The interrupt vector table is organized in 96 words of 32 bits. 32-bit, 16-bit, and 8-bit accesses are supported (when parity is disabled). Figure 14-7 shows the interrupt memory mapping. The table base address is FFF8 2000h.

Figure 14-7. VIM Interrupt Address Memory Map



NOTE: The interrupt vector table only has 96 entries, one phantom vector and 95 interrupt channels. Channel 95 does not have a dedicated vector and shall not be used.

There is one bit of parity per 32-bit ISR address. When a write is performed into the interrupt vector table, the parity is calculated for the 32-bit word and a parity bit is written into the corresponding parity region of interrupt vector table.

NOTE: Only 32-bit write/read access are allowed on interrupt vector table if parity is required. Non 32-bit access might result in parity errors.

When a read occurs from the CPU or VIM, the VIM calculates the parity from the data coming from the interrupt vector table and compares it to the parity stored in the table. The access of the data and the parity is performed in the same clock cycle.

If the parity bit does not match the calculated parity, a parity error is generated and the VIM stores the address of the error in the ADDERR register. The parity flag error (PARFLG) is set.

NOTE: The PARFLG register is only for bypassing the interrupt vector table in case of a parity fault. It should be used only to maintain the interrupt vector table bypassed. The checking of the parity fault should be done in the error signaling module (ESM) module where all parity errors are flagged.

Since the interrupt vector table may have an error, the FBPARERR register will provide to the VIC port, IRQVECREG and FIQVECREG, a fall-back address to an ISR that can restore the interrupt vector table content. The FBPARERR register should be set before initializing the interrupt in the interrupt vector table, to avoid branching to an unpredictable location.

The normal operation is restored when the PARFLG is cleared by the CPU. It is recommended to restore the content of the VIM before clearing the PARFLG.

The parity error signal is forwarded to the ESM.

14.4.2 Enabling and Controlling the VIM Parity

The polarity of VIM parity is controlled by the DEVCR1 register in the system module (address FFFF FFDCh). The parity enable is controlled by the PARCTL register. After reset, the parity is disabled.

Parity checking can be enabled by writing 0xA (1010b) in the PARENA[3:0] bit field of the PARCTL register. The default polarity is odd. The polarity can be changed to even by writing 5 (0101b) in the DEVPARSEL[3:0] bit field of the DEVCR1 register.

14.4.3 Interrupt Vector Table Initialization

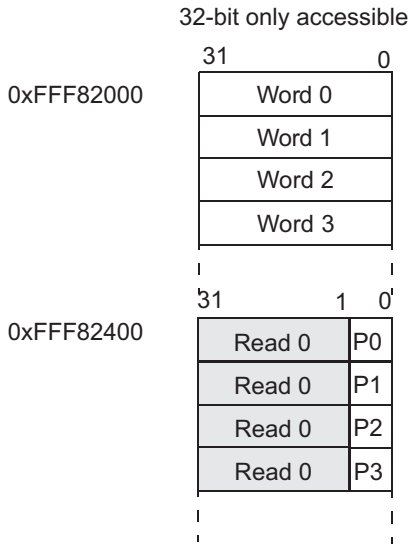
After reset, the interrupt vector table content including the parity bits is not initialized. Therefore, the CPU needs to initialize all of the interrupt addresses into the table, before enabling the corresponding interrupt channel. If parity is required, this initialization should be done after the parity functionality is enabled. In this way, the corresponding parity bit will be automatically updated. This initialization is only required when vectored interrupts are used, index interrupt management does not need the table to be initialized.

14.4.4 Interrupt Vector Table Parity Testing

To test the parity checking mechanism, the parity RAM allows manual insertion of faults. This option is implemented by the test bit in the PARCTL register. Once the bit is set, the parity bits are mapped to 0xFFF82400. After that, user can force faults into the parity bits. Finally, the parity error can be triggered by reading interrupt vector table (not parity bit) from VIM or CPU.

The interrupt vector table parity can also be verified by inserting faults into interrupt vector table. Once the VIM parity is disabled in system module, user can modify interrupt vector table without impacting the parity bit. After user re-enable interrupt vector table parity, the parity error can be triggered by reading interrupt vector table from VIM or CPU.

Figure 14-8. Parity Bit Mapping

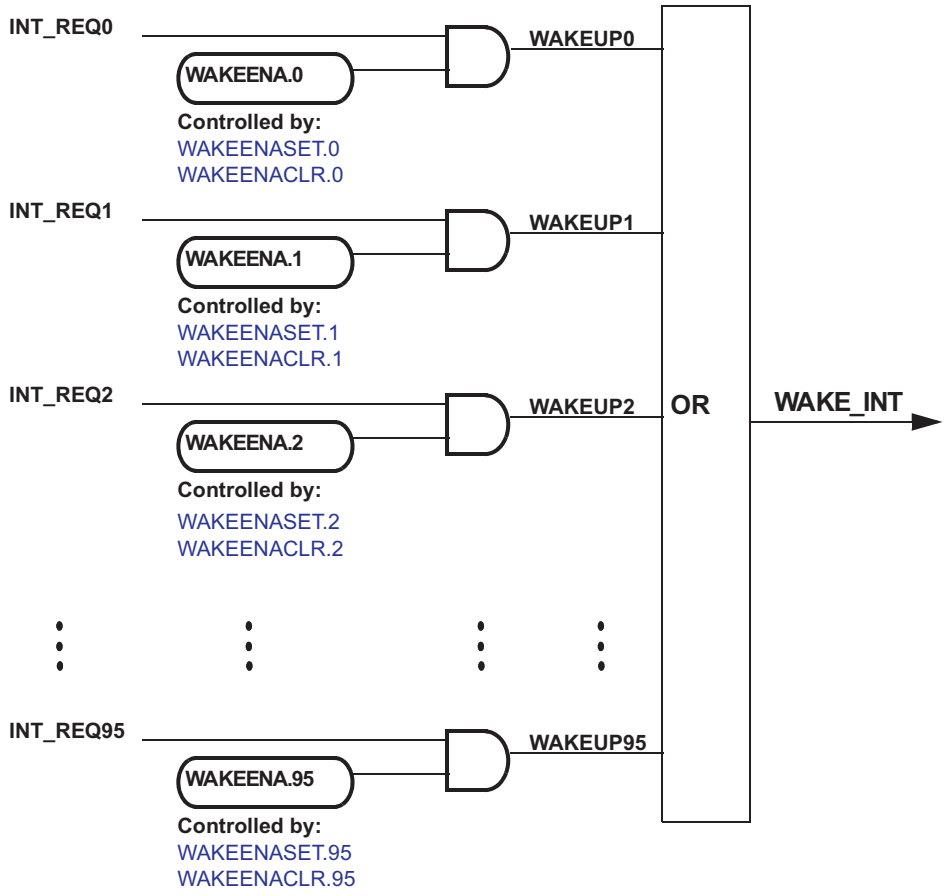


14.5 VIM Wakeup Interrupt

The wakeup interrupts are used to come out of low power mode (LPM). Any interrupt requests can be used to wake up the device. After reset, all interrupt requests are set to wake up from LPM. However, the VIM can mask unwanted interrupt lines for wake-up by using the WAKEENASET and WAKEENACLR register. The value in REQENASET/REQENACLR does NOT impact the wakeup interrupt.

As shown in Figure 14-9, the WAKEENASET and WAKEENACLR registers will enable/disable an interrupt for wake-up from low-power mode. All wake-up interrupts are “ORed” into a single signal WAKE_INT connected to the Global Clock Module.

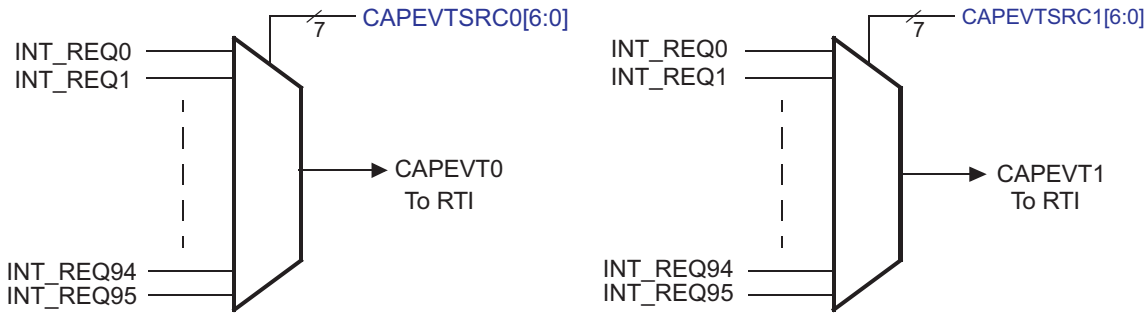
Figure 14-9. Detail of the IRQ Input



14.6 Capture Event Sources

The VIM can select any of the 96 interrupt requests to generate up to two capture events for the real-time interrupt (RTI) module (see Figure 14-10). The value in REQENASET/REQENACLR does NOT impact the capture event. Two registers (Section 14.8.16) are available, one for each capture event source.

Figure 14-10. Capture Event Sources



14.7 Examples

The following sections provide examples about the operation of the VIM.

14.7.1 Examples - Configure CPU To Receive Interrupts

Example 14-1 shows how to set the vector enable (VE) bit in the CP15 R1 register to enable the hardware vector interrupt. Example 14-2 shows how to enable/disable the IRQ and FIQ through CPSR. As a convention, the program who calls these subroutines shall preserve register R1 if needed. Example 14-2 can ONLY run in privileged mode. However, in USER mode, the application software can force the program into software interrupt by instruction SWI. Then, in the software interrupt service routine, user can write register SPSR, which is the copy of CPSR in this exception mode.

Example 14-1. Enable Hardware Vector Interrupt (IRQ Only)

```
_HW_Vec_Init
    MRC p15 ,#0 ,R1 ,c1 ,c0 ,#0
    ORR R1 ,R1 ,#0x01000000       ; Mask 0-31 bits except bit 24 in Sys
                                  ; Ctrl Reg of CORTEX-R4
    MCR p15 ,#0 ,R1 ,c1 ,c0 ,#0   ; Enable bit 24
    MOV PC, LR
```

Example 14-2. Enable/Disable IRQ/FIQ through CPSR

```
FIQENABLE .equ 0x40
IRQENABLE .equ 0x80
......
_Enable_Fiq
    MRS R1, CPSR
    BIC R1, R1, #FIQENABLE
    MSR CPSR, R1
    MOV PC, LR
......
_Disable_Irq
    MRS R1, CPSR
    ORR R1, R1, #IRQENABLE
    MSR CPSR, R1
    MOV PC, LR
......
_Enable_Irq
    MRS R1, CPSR
    BIC R1, R1, #IRQENABLE
    MSR CPSR, R1
    MOV PC, LR
```

14.7.2 Examples - Register Vector Interrupt and Index Interrupt Handling

Example 14-3 shows the configuration for the exception vectors in Register Vector Interrupt handling. After the interrupt is received by the CPU, the CPU branches to 0x18 (IRQ) or 0x1C (FIQ). The instruction placed here should be *LDR PC, [PC,#-0x1B0]*. The pending ISR address is written into the corresponding vector register (IRQVECREG for IRQ, FIQVECREG for FIQ). The CPU reads the content of the register and branches to the ISR.

Example 14-3. Exception Vector Configuration for VIM Vector

```
          .sect ".intvecs"
00000000h b _RESET             ; RESET interrupt
00000004h b _UNDEF_INST_INT    ; UNDEFINED INSTRUCTION interrupt
00000008h b _SW_INT            ; SOFTWARE interrupt
0000000Ch b _ABORT_PREF_INT    ; ABORT (PREFETCH) interrupt
00000010h b _ABORT_DATA_INT    ; ABORT (DATA) interrupt
00000014h b #-8                ; Reserved
00000018h ldr pc,[pc,#-0x1B0]  ; IRQ interrupt
0000001Ch ldr pc,[pc,#-0x1B0]  ; FIQ interrupt
```

NOTE: Program Counter (PC) always pointers two instructions beyond the current executed instruction. In this case, PC equals to '*0x18 or 0x1C + 0x08*'. The *LDR* instruction load the memory at '*PC - 0x1B0*', which is '*0x18 or 0x1C + 0x08 - 0x1B0* = 0xFFFFFE70 or 0xFFFFFE74'. These are the address of IRQVECREG and FIQVECREG, which store the pending ISR address.

Example 14-4 shows a fast response to the FIQ interrupt in Index Interrupt and can be applied to a system that has more than one channel assigned as a FIQ. It is built in Index Interrupt compatible with TMS470R1x legacy code.

Example 14-4. How to Respond to FIQ With Short Latency

```
          .sect ".intvecs"          ; Interrupt and exception vector sector
00000000h b _RESET                  ; RESET interrupt
00000004h b _UNDEF_INST_INT         ; UNDEFINED INSTRUCTION interrupt
00000008h b _SW_INT                 ; SOFTWARE interrupt
0000000Ch b _ABORT_PREF_INT         ; ABORT (PREFETCH) interrupt
00000010h b _ABORT_DATA_INT         ; ABORT (DATA) interrupt
00000014h b #-8                     ; Reserved
00000018h b _IRQ_ENTRY_0            ; IRQ interrupt
                                    ;*********************************
                                    ; INTERRUPT PROCESSING AREA
                                    ;*********************************
0000001Ch ldrb R8, [PC,#-0x21d]     ; FIQ INTERRUPT ENTRY
                                    ; R8 used to get the FIQ index
                                    ; with address pointer to the
                                    ; first FIQ banked register
00000020h ldr PC, [PC, R8, LSL#2]   ; Branch to the indexed interrupt
                                    ; routine. The prefetch
                                    ; operation causes the PC to be 2
                                    ; words (8 bytes) ahead of the
                                    ; current instruction, so
                                    ; pointing to _INT_TABLE.
00000024h nop                       ; Required due to pipeline.
                                    ;=================================
00000028h _INT_TABLE                ; FIQ INTERRUPT DISPATCH
                                    ;=================================
0000002Ch .word _FIQ_TABLE          ; beginning of FIQ Dispatch
00000030h .word _ISR1               ; dispatch to interrupt routine 1
00000034h .word _ISR2               ; dispatch to interrupt routine 2
          .
          .
```

Another way to improve the FIQ latency is to assign only one channel to the FIQ interrupt and to map the ISR code corresponding to this channel directly starting at 0x1C.

NOTE: When the CPU is in vector-enabled mode, Example 14-3 and Example 14-4 are still valid. The difference is that the CPU will not read from the 0x18 location during IRQ interrupt, but will jump directly to the corresponding ISR routine.

14.8 VIM Control Registers

This section details the VIM module registers, summarized in Table 14-1.

Each register begins on a word boundary. All registers are 32-bit, 16-bit, and 8-bit accessible for read and write. Write is only possible in privilege mode. The base address of the control registers is FFFF FE00h. The base address of the parity-related VIM registers is FFFF FD00h. The address locations not listed are reserved.

Table 14-1. VIM Control Registers

Offset	Acronym	Register Description	Section
		Parity-related Registers	
ECh	PARFLG	Interrupt Vector Table Parity Flag Register	Section 14.8.1
F0h	PARCTL	Interrupt Vector Table Parity Control Register	Section 14.8.2
F4h	ADDERR	Address Parity Error Register	Section 14.8.3
F8h	FBPARERR	Fall-Back Address Parity Error Register	Section 14.8.4
		Control Registers	
00h	IRQINDEX	IRQ Index Offset Vector Register	Section 14.8.6
04h	FIQINDEX	FIQ Index Offset Vector Register	Section 14.8.7
10h	FIRQPR0	FIQ/IRQ Program Control Register 0	Section 14.8.8
14h	FIRQPR1	FIQ/IRQ Program Control Register 1	Section 14.8.8
18h	FIRQPR2	FIQ/IRQ Program Control Register 2	Section 14.8.8
20h	INTREQ0	Pending Interrupt Read Location Register 0	Section 14.8.9
24h	INTREQ1	Pending Interrupt Read Location Register 1	Section 14.8.9
28h	INTREQ2	Pending Interrupt Read Location Register 2	Section 14.8.9
30h	REQENASET0	Interrupt Enable Set Register 0	Section 14.8.10
34h	REQENASET1	Interrupt Enable Set Register 1	Section 14.8.10
38h	REQENASET2	Interrupt Enable Set Register 2	Section 14.8.10
40h	REQENACLR0	Interrupt Enable Clear Register 0	Section 14.8.11
44h	REQENACLR1	Interrupt Enable Clear Register 1	Section 14.8.11
48h	REQENACLR2	Interrupt Enable Clear Register 2	Section 14.8.11
50h	WAKEENASET0	Wake-up Enable Set Register 0	Section 14.8.12
54h	WAKEENASET1	Wake-up Enable Set Register 1	Section 14.8.12
58h	WAKEENASET2	Wake-up Enable Set Register 2	Section 14.8.12
60h	WAKEENACLR0	Wake-up Enable Clear Register 0	Section 14.8.13
64h	WAKEENACLR1	Wake-up Enable Clear Register 1	Section 14.8.13
68h	WAKEENACLR2	Wake-up Enable Clear Register 2	Section 14.8.13
70h	IRQVECREG	IRQ Interrupt Vector Register	Section 14.8.14
74h	FIQVECREG	FIQ Interrupt Vector Register	Section 14.8.15
78h	CAPEVT	Capture Event Register	Section 14.8.16
80h-DCh	CHANCTRL	VIM Interrupt Control Register	Section 14.8.17

14.8.1 Interrupt Vector Table Parity Flag Register (PARFLG)

Figure 14-11 and Table 14-2 describe this register.

Figure 14-11. Interrupt Vector Table Parity Flag Register (PARFLG) [offset = ECh]

31	16
Reserved	
R-0	

15	1	0
Reserved		PARFLG
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 14-2. Interrupt Vector Table Parity Flag Register (PARFLG) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	PARFLG		The PARFLG indicates that a parity error has been found and that the Interrupt Vector Table is bypassed. The resulting vector of any IRQ/FRQ interrupt is then the value contained in the FBPARERR register until this bit has been cleared.
		0	Read: No parity error has occurred. Write: A write to this bit has no effect.
		1	Read: A parity error has occurred and the Interrupt Vector Table is bypassed. Write: The PARFLG is cleared and the interrupt vector can be read from the Interrupt Vector Table.

14.8.2 Interrupt Vector Table Parity Control Register (PARCTL)

Figure 14-12. Interrupt Vector Table Parity Control Register (PARCTL) [offset = F0h]

31	16
Reserved	
R-0	

15	9	8	7	4	3	0
Reserved		TEST	Reserved		PARENA	
R-0		R/WP-0	R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 14-3. Interrupt Vector Table Parity Control Register (PARCTL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Read returns 0. Writes have no effect.
8	TEST		This bit maps the parity bits into the Interrupt Vector Table frame to make them accessible by the CPU.
		0	Parity bits are not memory mapped.
		1	Parity bits are memory mapped.
7-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	PARENA		VIM parity enable.
		5h	The VIM parity is disabled.
		All Others	The VIM parity is enabled.
			Note: To avoid soft error to disable VIM parity checking, it is recommended to write Ah to enable parity checking.

14.8.3 Address Parity Error Register (ADDERR)

The address parity error register gives the address of the first parity error location.

NOTE: No computation is needed when reading the complete register to retrieve the address in the Interrupt Vector Table.

This register will never be reset by a power-on reset nor any other reset source.

Figure 14-13. Address Parity Error Register (ADDERR) [offset = F4h]

31	16
Interrupt Vector Table offset	
R-FFF8h	

15	9	8	2	1	0
Interrupt Vector Table offset		ADDERR		Word offset	
R-0010 000b		R-x		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 14-4. Address Parity Error Register (ADDERR) Field Descriptions

Bit	Field	Description
31-9	Interrupt Vector Table offset	Interrupt Vector Table offset. Reads are always FFF8 2xxxh; writes have no effect.
8-2	ADDERR	Address parity error register. This register gives the address of the first encountered parity error since the flag has been clear. Subsequent parity errors will not update this register until the PARFLG register has been cleared. **Note: This register is valid only when PARFLG is set (see Section 14.8.1).**
1-0	Word offset	Word offset. Reads are always 0; writes have no effect.

14.8.4 Fall-Back Address Parity Error Register (FBPARERR)

This register provides a fall-back address to the VIM if a parity error has occurred in the Interrupt Vector Table. Figure 14-14 and Table 14-5 describe this register.

NOTE: This register will never be reset by a power-on reset nor any other reset source.

Figure 14-14. Fall-Back Address Parity Error Register (FBPARERR) [offset = F8h]

31	16
FBPARERR	
R/WP-x	

15	0
FBPARERR	
R/WP-x	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset; x = Indeterminate

Table 14-5. Fall Back Address Parity Error Register (FBPARERR) Field Descriptions

Bit	Field	Value	Description
31-0	FBPARERR	0-FFFF FFFFh	Fall back address parity error. This register is used by the VIM if the Interrupt Vector Table has been corrupted. The contents of the IRQVECREG and FIQVECREG registers will reflect the value programmed in FBPARERR. The value provided to the VIC port will also reflect FBPARERR until the PARFLG register has been cleared. This register provides the address of the ISR that will restore the integrity of the Interrupt Vector Table.

14.8.5 VIM Offset Vector Registers

The VIM offset register provides the user with the numerical index value that represents the pending interrupt with the highest precedence. The register IRQINDEX holds the index to the highest priority IRQ interrupt; the register FIQINDEX holds the index to the highest priority FIQ interrupt. The index can be used to locate the interrupt routine in a dispatch table, as shown in Table 14-6.

Table 14-6. Interrupt Dispatch

IRQINDEX / FIQINDEX Register Bit Field	Highest Priority Pending Interrupt Enabled
0x00	No interrupt
0x01	Channel 0
:	:
0x5F	Channel 94
0x60	Channel 95

NOTE: Channel 95 has no dedicated interrupt vector table entry. Therefore, Channel 95 shall NOT be used in application.

The VIM offset registers are read only. They are updated continuously by the VIM. When an interrupt is serviced, the offset vectors show the index for the next highest pending interrupt or 0x0 if no interrupt is pending.

14.8.6 IRQ Index Offset Vector Register (IRQINDEX)

The IRQ offset register provides the user with the numerical index value that represents the pending IRQ interrupt with the highest priority. Figure 14-15 and Table 14-7 describe this register.

Figure 14-15. IRQ Index Offset Vector Register (IRQINDEX) [offset = 00h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		IRQINDEX	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 14-7. IRQ Index Offset Vector Register (IRQINDEX) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	IRQINDEX	0-FFh	IRQ index vector. The least-significant bits represent the index of the IRQ pending interrupt with the highest precedence, as shown in Table 14-6. When no interrupts are pending, the least-significant byte of IRQINDEX is 0. **Note:** A read of register IRQINDEX or IRQVECREG will cause IRQINDEX / IRQVECREG to reflect the index/ISR address for the next highest-priority pending IRQ interrupt. In case there is no other interrupt pending, the IRQINDEX will read 0x00 and the IRQVECREG register will read the phantom interrupt address.

14.8.7 FIQ Index Offset Vector Registers (FIQINDEX)

The FIQINDEX register provides the user with a numerical index value that represents the pending FIQ interrupt with the highest priority. Figure 14-16 and Table 14-8 describe this register.

Figure 14-16. FIQ Index Offset Vector Register (FIQINDEX) [offset = 04h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		FIQINDEX	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 14-8. FIQ Index Offset Vector Register (FIQINDEX) Field Descriptions

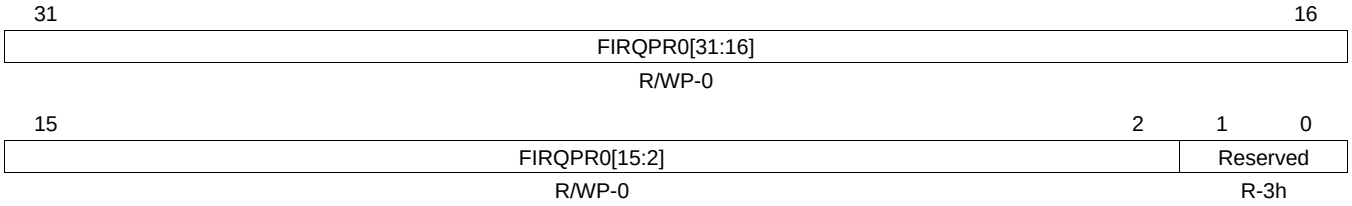
Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	FIQINDEX	0-FFh	FIQ index offset vector. The least-significant bits represent the index of the FIQ pending interrupt with the highest precedence, as shown in Table 14-6. When no interrupts are pending, the least-significant byte of FIQINDEX is 0x00. **Note:** A read of register FIQINDEX or FIQVECREG will cause FIQINDEX / FIQVECREG to reflect the index/ISR address for the next highest-priority pending FIQ interrupt. In case there is no other interrupt pending, the FIQINDEX will read 0x00 and the FIQVECREG register will read the phantom interrupt address.

14.8.8 FIQ/IRQ Program Control Registers (FIRQPR[0:2])

The FIQ/IRQ program control registers (FIRQPRx) determine whether a given interrupt request will be either FIQ or IRQ. Figure 14-17, Figure 14-18, Figure 14-19 and Table 14-9 describe these registers.

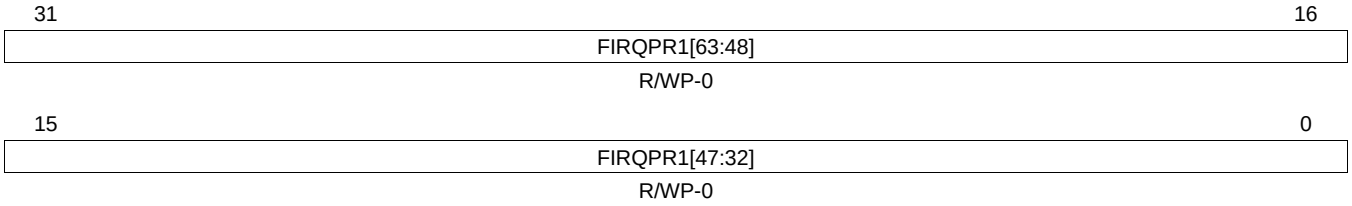
NOTE: Channel 0 and 1 are FIQ only, not impacted by this register.

Figure 14-17. FIQ/IRQ Program Control Register 0 (FIRQPR0) [offset = 10h]



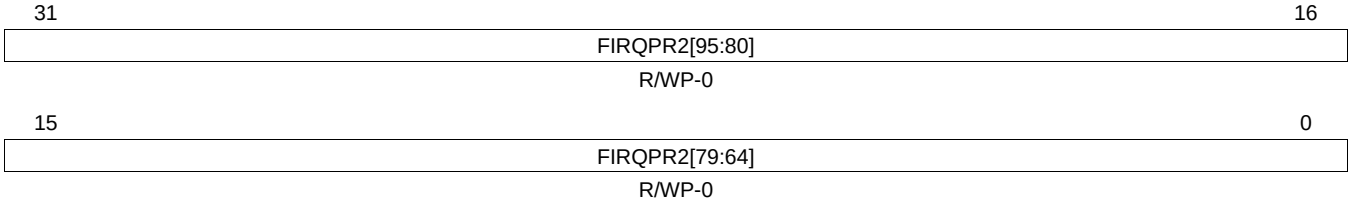
LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-18. FIQ/IRQ Program Control Register 1 (FIRQPR1) [offset = 14h]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-19. FIQ/IRQ Program Control Register 2 (FIRQPR2) [offset = 18h]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 14-9. FIQ/IRQ Program Control Registers (FIRQPRx) Field Descriptions

Bit	Field	Value	Description
95-2	FIRQPRx[95:2]		FIQ/IRQ program control bits. These bits determine whether an interrupt request from a peripheral is of type FIQ or IRQ. Bit FIRQPRx[95:2] corresponds to request channel[95:2].
		0	Interrupt request is of IRQ type.
		1	Interrupt request is of FIQ type.
1-0	Reserved	3h	Read only. Writes have no effect.

14.8.9 Pending Interrupt Read Location Registers (INTREQ[0:2])

The pending interrupt registers (INTREQx) give the pending interrupt requests. The register is updated every vbus clock cycle. Figure 14-20, Figure 14-21, Figure 14-22 and Table 14-10 describe this register.

Figure 14-20. Pending Interrupt Read Location Register 0 (INTREQ0) [offset = 20h]

31	16
INTREQ0[31:16]	
R/W1CP-0	
15	0
INTREQ0[15:0]	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Figure 14-21. Pending Interrupt Read Location Register 1 (INTREQ1) Register [offset = 24h]

31	16
INTREQ1[63:48]	
R/W1CP-0	
15	0
INTREQ1[47:32]	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Figure 14-22. Pending Interrupt Read Location Register 2 (INTREQ2) Register [offset = 28h]

31	16
INTREQ2[95:80]	
R/W1CP-0	
15	0
INTREQ2[79:64]	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 14-10. Pending Interrupt Read Location Registers (INTREQx) Field Descriptions

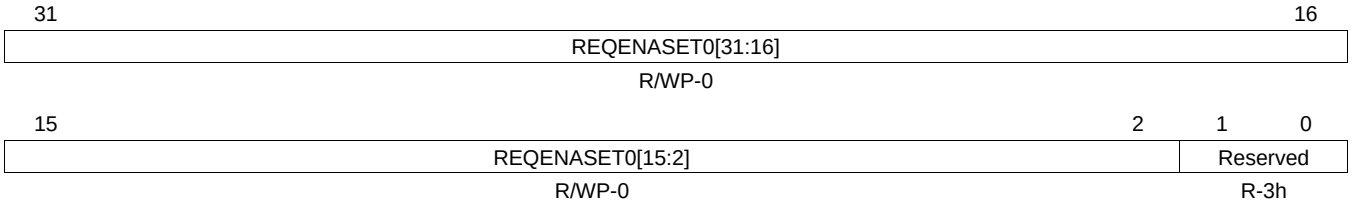
Bit	Field	Value	Description
95-0	INTREQx[95:0]		Pending interrupt bits. These bits determine whether an interrupt request is pending for the request channel between 0 and 95. The interrupt ENABLE register does not affect the value of the interrupt pending bit. Bit INTREQx[95:0] corresponds to request channel[95:0].
			User and Privilege Mode read:
		0	No interrupt event has occurred.
		1	An interrupt is pending.
			Privilege Mode write only:
		0	Writing 0 has no effect.
		1	Clears the interrupt pending status flag. This write-clear functionality is intended to allow clearing those interrupts which have been signaled to VIM before enabling the interrupt channel, if they are undesired.

14.8.10 Interrupt Enable Set Registers (REQENASET[0:2])

The interrupt enable set registers (REQENASETx) selectively enables individual request channels. Figure 14-23, Figure 14-24, Figure 14-25 and Table 14-11 describe these registers.

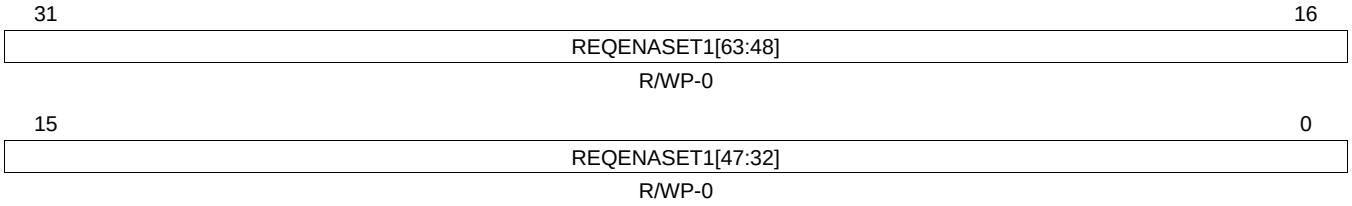
NOTE: Channel 0 and 1 are always enabled, not impacted by this register.

Figure 14-23. Interrupt Enable Set Register 0 (REQENASET0) [offset = 30h]



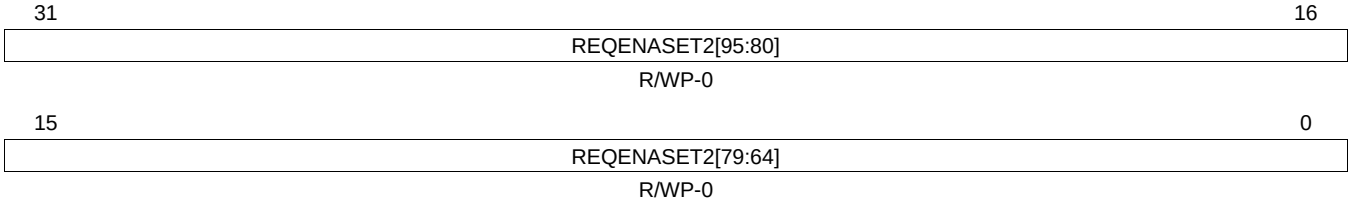
LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-24. Interrupt Enable Set Register 1 (REQENASET1) [offset = 34h]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-25. Interrupt Enable Set Register 2 (REQENASET2) [offset = 38h]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 14-11. Interrupt Enable Set Registers (REQENASETx) Field Descriptions

Bit	Field	Value	Description
95-2	REQENASETx[95:2]		Request enable set bits. This vector determines whether the interrupt request channel is enabled. Bit REQENASETx[95:2] corresponds to request channel[95:2].
		0	Read: Interrupt request channel is disabled. Write: A write of 0 has no effect.
		1	Read or Write: The interrupt request channel is enabled.
1-0	Reserved	3h	Read only. Writes have no effect.

14.8.11 Interrupt Enable Clear Registers (REQENACLR[0:2])

The interrupt enable clear registers (REQENACLRx) selectively disables individual request channels. Figure 14-26, Figure 14-27, Figure 14-28 and Table 14-12 describe these registers.

NOTE: Channel 0 and 1 are always enabled, not impacted by this register.

Figure 14-26. Interrupt Enable Clear Register 0 (REQENACLR0) [offset = 40h]

31	16
REQENACLR0[31:16]	
R/WP-0	

15	2	1	0
REQENACLR0[15:2]		Reserved	
R/WP-0		R-3h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-27. Interrupt Enable Clear Register 1 (REQENACLR1) [offset = 44h]

31	16
REQENACLR1[63:48]	
R/WP-0	

15	0
REQENACLR1[47:32]	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-28. Interrupt Enable Clear Register 2 (REQENACLR2) [offset = 48h]

31	16
REQENACLR2[95:80]	
R/WP-0	

15	0
REQENACLR2[79:64]	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 14-12. Interrupt Enable Clear Registers (REQENACLRx) Field Descriptions

Bit	Field	Value	Description
95-2	REQENACLRx[95:2]		Request enable clear bits. This vector determines whether the interrupt request channel is enabled. Bit REQENACLRx[95:2] corresponds to request channel[95:2].
		0	Read: Interrupt request channel is disabled. Write: A write of 0 has no effect.
		1	Read: The interrupt request channel is enabled. Write: The interrupt request channel is disabled.
1-0	Reserved	3h	Read only. Writes have no effect.

14.8.12 Wake-Up Enable Set Registers (WAKEENASET[0:2])

The wake-up enable set registers (WAKEENASETx) selectively enables individual wake-up interrupt request lines. Figure 14-29, Figure 14-30, Figure 14-31 and Table 14-13 describe these registers.

Figure 14-29. Wake-Up Enable Set Register 0 (WAKEENASET0) [offset = 50h]

31	16
WAKEENASET0[31:16]	
R/WP-FFFFh	
15	0
WAKEENASET0[15:0]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-30. Wake-Up Enable Set Register 1 (WAKEENASET1) [offset = 54h]

31	16
WAKEENASET1[63:48]	
R/WP-FFFFh	
15	0
WAKEENASET1[47:32]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-31. Wake-Up Enable Set Register 2 (WAKEENASET2) [offset = 58h]

31	16
WAKEENASET2[95:80]	
R/WP-FFFFh	
15	0
WAKEENASET2[79:64]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

Table 14-13. Wake-Up Enable Set Registers (WAKEENASETx) Field Descriptions

Bit	Field	Value	Description
95-0	WAKEENASETx[95:0]		Wake-up enable set bits. This vector determines whether the wake-up interrupt line is enabled. Bit WAKEENASETx[95:0] corresponds to interrupt request channel[95:0].
		0	Read: Interrupt request channel is disabled. Write: A write of 0 has no effect.
		1	Read or Write: The interrupt request channel is enabled.

14.8.13 Wake-Up Enable Clear Registers (WAKEENACLR[0:2])

The wake-up enable clear registers (WAKEENACLRx) selectively disables individual wake-up interrupt request lines. Figure 14-32, Figure 14-33, Figure 14-34 and Table 14-14 describe these registers.

Figure 14-32. Wake-Up Enable Clear Register 0 (WAKEENACLR0) [offset = 60h]

31	16
WAKEENACLR0[31:16]	
R/WP-FFFFh	
15	0
WAKEENACLR0[15:0]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-33. Wake-Up Enable Clear Register 1 (WAKEENACLR1) [offset = 64h]

31	16
WAKEENACLR1[63:48]	
R/WP-FFFFh	
15	0
WAKEENACLR1[47:32]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

Figure 14-34. Wake-Up Enable Clear Register 2 (WAKEENACLR2) [offset = 68h]

31	16
WAKEENACLR2[95:80]	
R/WP-FFFFh	
15	0
WAKEENACLR2[79:64]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

Table 14-14. Wake-Up Enable Clear Registers (WAKEENACLRx) Field Descriptions

Bit	Field	Value	Description
95-0	WAKEENACLRx[95:0]		Wake-up enable clear bits. This vector determines whether the wake-up interrupt line is enabled. Bit WAKEENACLRx[95:0] corresponds to interrupt request channel[95:0].
		0	Read: Wake-up interrupt channel is disabled. Write: A write of 0 has no effect.
		1	Read: The wake-up interrupt channel is enabled. Write: The wake-up interrupt channel is disabled.

14.8.14 IRQ Interrupt Vector Register (IRQVECREG)

The interrupt vector register gives the address of the enabled and active IRQ interrupt. Figure 14-35 and Table 14-15 describe these registers.

Figure 14-35. IRQ Interrupt Vector Register (IRQVECREG) [offset = 70h]

31	16
IRQVECREG	
R-0	

15	0
IRQVECREG	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 14-15. IRQ Interrupt Vector Register (IRQVECREG) Field Descriptions

Bit	Field	Value	Description
31-0	IRQVECREG	From Section 14.4	IRQ interrupt vector register. This vector gives the address of the ISR with the highest pending IRQ request. The CPU reads the address and branches to this location. **Note:** A read of register IRQINDEX or IRQVECREG will cause IRQINDEX / IRQVECREG to reflect the index/ISR address for the next highest-priority pending IRQ interrupt. In case there is no other interrupt pending, the IRQINDEX will read 0x00 and the IRQVECREG register will read the phantom interrupt address.

14.8.15 FIQ Interrupt Vector Register (FIQVECREG)

The interrupt vector register gives the address of the enabled and active FIQ interrupt. Figure 14-36 and Table 14-16 describe these registers.

Figure 14-36. IRQ Interrupt Vector Register (FIQVECREG) [offset = 74h]

31	16
FIQVECREG	
R-0	

15	0
FIQVECREG	
R-0	

LEGEND: R = Read only; -*n* = value after reset; X = Unknown

Table 14-16. FIQ Interrupt Vector Register (FIQVECREG) Field Descriptions

Bit	Field	Value	Description
31-0	FIQVECREG	From Section 14.4	FIQ interrupt vector register. This vector gives the address of the ISR with the highest pending FIQ request. The CPU reads the address and branches to this location. **Note:** A read of register FIQINDEX or FIQVECREG will cause FIQINDEX / FIQVECREG to reflect the index/ISR address for the next highest-priority pending FIQ interrupt. In case there is no other interrupt pending, the FIQINDEX will read 0x00 and the FIQVECREG register will read the phantom interrupt address.

14.8.16 Capture Event Register (CAPEVT)

Figure 14-37 and Table 14-17 describe this register.

Figure 14-37. Capture Event Register (CAPEVT) [offset = 78h]

31	23	22	16
Reserved		CAPEVTSRC1	
R-U		R/W-0	

15	7	6	0
Reserved		CAPEVTSRC0	
R-U		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; U = Undefined; -*n* = value after reset

Table 14-17. Capture Event Register (CAPEVT) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reads are indeterminate and writes have no effect.
22-16	CAPEVTSRC1		Capture event source 1 mapping control. These bits determine which interrupt request maps to the capture event source 1 of the RTI:
		0	Interrupt request 0.
		1h	Interrupt request 1.
		:	:
		5Fh	Interrupt request 95.
15-7	Reserved	0	Reads are indeterminate and writes have no effect.
6-0	CAPEVTSRC0		Capture event source 0 mapping control. These bits determine which interrupt request maps to the capture event source 0 of the RTI:
		0	Interrupt request 0.
		1h	Interrupt request 1.
		:	:
		5Fh	Interrupt request 95.

14.8.17 VIM Interrupt Control Registers (CHANCTRL[0:23])

Twenty-four interrupt control registers control the 96 interrupt channels of the VIM. Each register controls four interrupt channels: each of them is indexed from 0 to 95. Table 14-18 shows the organization of all the registers and the reset value of each. Each four fields of the register has been named with a generic index that refers to the detailed register organization. Figure 14-38 and Table 14-19 describe these registers.

Table 14-18. Interrupt Control Registers Organization

Address	Register Acronym	Register Field 31:24 CHANMAPx$_0$	Register Field 23:16 CHANMAPx$_1$	Register Field 15:8 CHANMAPx$_2$	Register Field 7:0 CHANMAPx$_3$	Reset Value
FFFF FE80h	CHANCTRL0	CHANMAP0	CHANMAP1	CHANMAP2	CHANMAP3	0001 0203h
FFFF FE84h	CHANCTRL1	CHANMAP4	CHANMAP5	CHANMAP6	CHANMAP7	0405 0607h
:	:	:	:	:	:	:
FFFF FED8h	CHANCTRL22	CHANMAP88	CHANMAP89	CHANMAP90	CHANMAP91	5859 5A5Bh
FFFF FEDCh	CHANCTRL23	CHANMAP92	CHANMAP93	CHANMAP94	CHANMAP95	5C5D 5E5Fh

NOTE: CHANMAP0 and CHANMAP1 are not programable. CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1.

Do NOT write any value other than 0x5F to CHANMAP95. Channel 95 is reserved because no interrupt vector table entry supports this channel.

Figure 14-38. Interrupt Control Registers (CHANCTRL[0:23]) [offset = 80h-DCh]

31	30 24	23	22 16
Rsvd	CHANMAPx$_0$	Rsvd	CHANMAPx$_1$
R-U	R/W-n	R-U	R/W-n

15	14 8	7	6 0
Rsvd	CHANMAPx$_2$	Rsvd	CHANMAPx$_3$
R-U	R/W-n	R-U	R/W-n

LEGEND: R/W = Read/Write; R = Read only; U = Undefined; -*n* = value after reset (see Table 14-18)

Table 14-19. Interrupt Control Registers (CHANCTRLx) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reads are indeterminate and writes have no effect.
30-24	CHANMAPx$_0$		CHANMAPx$_0$(6-0). Interrupt CHANx$_0$ mapping control. These bits determine which interrupt request the priority channel CHANx$_0$ maps to:
		0	Read: Interrupt request 0 maps to channel priority CHANx$_0$. Write: The default value of this bit after reset is given in Table 14-18 . The channel priority CHANx$_0$ is set with the interrupt request.
		1h	Read: Interrupt request 1 maps to channel priority CHANx$_0$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority CHANx$_0$ is set with the interrupt request.
		:	:
		5Fh	Read: Interrupt request 95 maps to channel priority CHANx$_0$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority CHANx$_0$ is set with the interrupt request.
23	Reserved	0	Reads are indeterminate and writes have no effect.

Table 14-19. Interrupt Control Registers (CHANCTRLx) Field Descriptions (continued)

Bit	Field	Value	Description
22-16	$CHANMAPx_1$		$CHANMAPx_1$(6-0). Interrupt $CHANx_1$ mapping control. These bits determine which interrupt request the priority channel $CHANx_1$ maps to:
		0	Read: Interrupt request 0 maps to channel priority $CHANx_1$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_1$ is set with the interrupt request.
		1h	Read: Interrupt request 1 maps to channel priority $CHANx_1$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_1$ is set with the interrupt request.
		:	:
		5Fh	Read: Interrupt request 95 maps to channel priority $CHANx_1$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_1$ is set with the interrupt request.
15	Reserved	0	Reads are indeterminate and writes have no effect.
14-8	$CHANMAPx_2$		$CHANMAPx_2$(6-0). Interrupt $CHANx_2$ mapping control. These bits determine which interrupt request the priority channel $CHANx_2$ maps to:
		0	Read: Interrupt request 0 maps to channel priority $CHANx_2$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_2$ is set with the interrupt request.
		1h	Read: Interrupt request 1 maps to channel priority $CHANx_2$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_2$ is set with the interrupt request.
		:	:
		5Fh	Read: Interrupt request 95 maps to channel priority $CHANx_2$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_2$ is set with the interrupt request.
7	Reserved	0	Reads are indeterminate and writes have no effect.
6-0	$CHANMAPx_3$		$CHANMAPx_3$(6-0). Interrupt $CHANx_3$ mapping control. These bits determine which interrupt request the priority channel $CHANx_3$ maps to:
		0	Read: Interrupt request 0 maps to channel priority $CHANx_3$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_3$ is set with the interrupt request.
		1h	Read: Interrupt request 1 maps to channel priority $CHANx_3$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_3$ is set with the interrupt request.
		:	:
		5Fh	Read: Interrupt request 95 maps to channel priority $CHANx_3$. Write: The default value of this bit after reset is given in Table 14-18. The channel priority $CHANx_3$ is set with the interrupt request.

Enhanced Quadrature Encoder Pulse (eQEP) Module

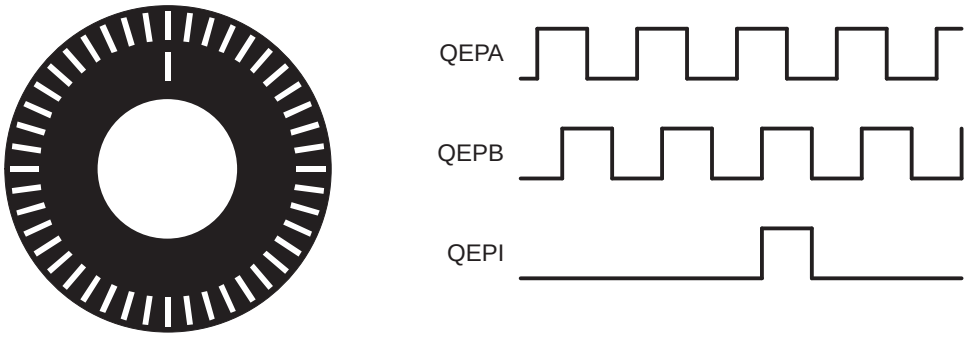
The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system.

Topic **Page**

15.1 Introduction

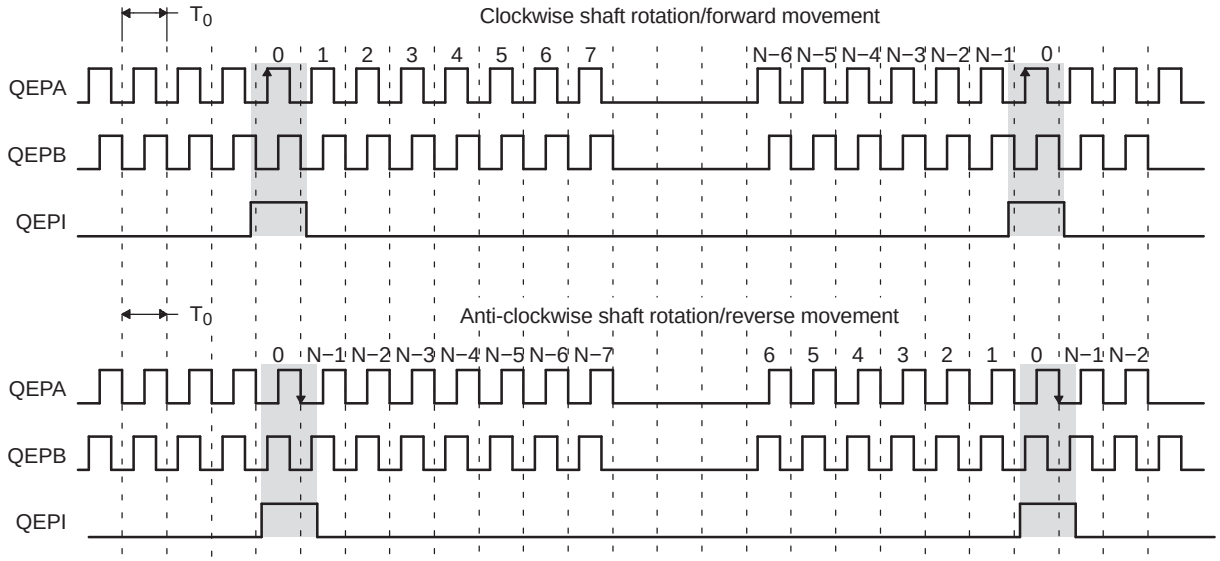
A single track of slots patterns the periphery of an incremental encoder disk, as shown in Figure 15-1. These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark/light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position, and zero reference

Figure 15-1. Optical Encoder Disk



To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is realized with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90° out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and vise versa as shown in Figure 15-2.

Figure 15-2. QEP Encoder Output Signal for Forward/Reverse Movement

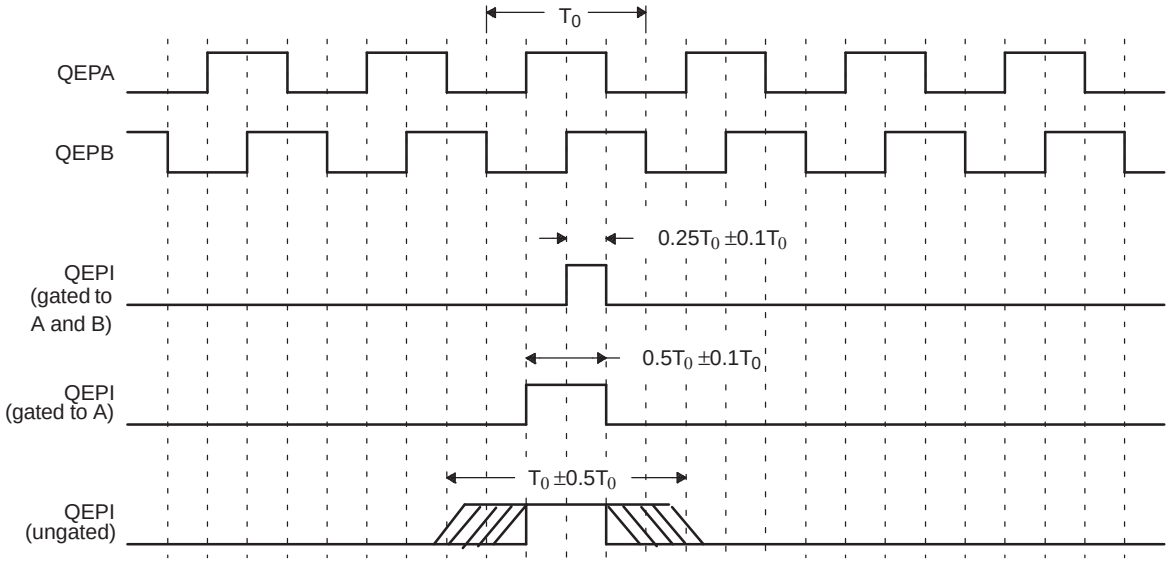


Legend: N = lines per revolution

The encoder wheel typically makes one revolution for every revolution of the motor or the wheel may be at a geared rotation ratio with respect to the motor. Therefore, the frequency of the digital signal coming from the QEPA and QEPB outputs varies proportionally with the velocity of the motor. For example, a 2000-line encoder directly coupled to a motor running at 5000 revolutions per minute (rpm) results in a frequency of 166.6 KHz, so by measuring the frequency of either the QEPA or QEPB output, the processor can determine the velocity of the motor.

Quadrature encoders from different manufacturers come with two forms of index pulse (gated index pulse or ungated index pulse) as shown in Figure 15-3. A nonstandard form of index pulse is ungated. In the ungated configuration, the index edges are not necessarily coincident with A and B signals. The gated index pulse is aligned to any of the four quadrature edges and width of the index pulse and can be equal to a quarter, half, or full period of the quadrature signal.

Figure 15-3. Index Pulse Example



Some typical applications of shaft encoders include robotics and even computer input in the form of a mouse. Inside your mouse you can see where the mouse ball spins a pair of axles (a left/right, and an up/down axle). These axles are connected to optical shaft encoders that effectively tell the computer how fast and in what direction the mouse is moving.

General Issues: Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity may be written as:

$$v(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (26)$$

$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (27)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

t(k): Time instant "k"

t(k-1): Time instant "k-1"

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement.

Equation 26 is the conventional approach to velocity estimation and it requires a time base to provide unit time event for velocity calculation. Unit time is basically the inverse of the velocity calculation rate.

The encoder count (position) is read once during each unit time event. The quantity [x(k) - x(k-1)] is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant 1/T (where T is the constant time between unit time events and is known in advance).

Estimation based on Equation 26 has an inherent accuracy limit directly related to the resolution of the position sensor and the unit time period T. For example, consider a 500-line per revolution quadrature encoder with a velocity calculation rate of 400 Hz. When used for position the quadrature encoder gives a four-fold increase in resolution, in this case, 2000 counts per revolution. The minimum rotation that can be detected is therefore 0.0005 revolutions, which gives a velocity resolution of 12 rpm when sampled at 400 Hz. While this resolution may be satisfactory at moderate or high speeds, for example, 1% error at 1200 rpm, it would clearly prove inadequate at low speeds. In fact, at speeds below 12 rpm, the speed estimate would erroneously be zero much of the time.

At low speed, Equation 27 provides a more accurate approach. It requires a position sensor that outputs a fixed interval pulse train, such as the aforementioned quadrature encoder. The width of each pulse is defined by motor speed for a given sensor resolution. Equation 27 can be used to calculate motor speed by measuring the elapsed time between successive quadrature pulse edges. However, this method suffers from the opposite limitation, as does Equation 26. A combination of relatively large motor speeds and high sensor resolution makes the time interval ΔT small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use Equation 27 at low speed and have the application software switch over to Equation 26 when the motor speed rises above some specified threshold.

15.2 Description

This section provides the eQEP inputs, memory map, and functional description.

15.2.1 eQEP Inputs

The eQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or 0 marker), and a strobe input.

- *QEPA/XCLK and QEPB/XDIR*

  These two pins can be used in quadrature-clock mode or direction-count mode.

  – *Quadrature-clock Mode*

    The eQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase whose phase relationship is used to determine the direction of rotation of the input shaft and number of eQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and vice versa. The quadrature decoder uses these two inputs to generate quadrature-clock and direction signals.

  – *Direction-count Mode*

    In direction-count mode, direction and clock signals are provided directly from the external source. Some position encoders have this type of output instead of quadrature output. The QEPA pin provides the clock input and the QEPB pin provides the direction input.

- *eQEPI: Index or Zero Marker*

  The eQEP encoder uses an index signal to assign an absolute start position from which position information is incrementally encoded using quadrature pulses. This pin is connected to the index output of the eQEP encoder to optionally reset the position counter for each revolution. This signal can be used to initialize or latch the position counter on the occurrence of a desired event on the index pin.

- QEPS: *Strobe Input*

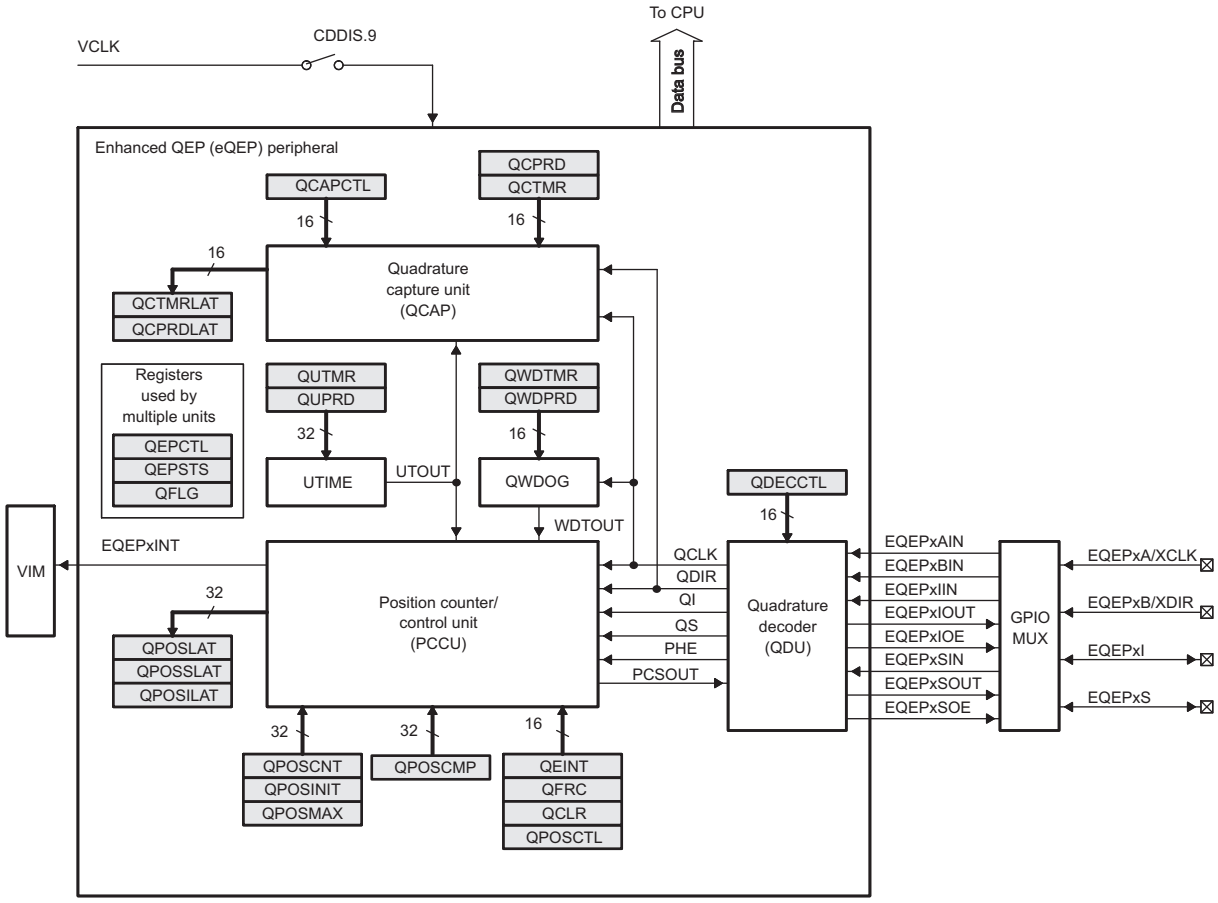
  This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.

15.2.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in Figure 15-4):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG)

Figure 15-4. Functional Block Diagram of the eQEP Peripheral



15.2.3 eQEP Memory Map

Table 15-1 lists the registers with their memory locations, sizes, and reset values. See Section 15.9 for details of these registers.

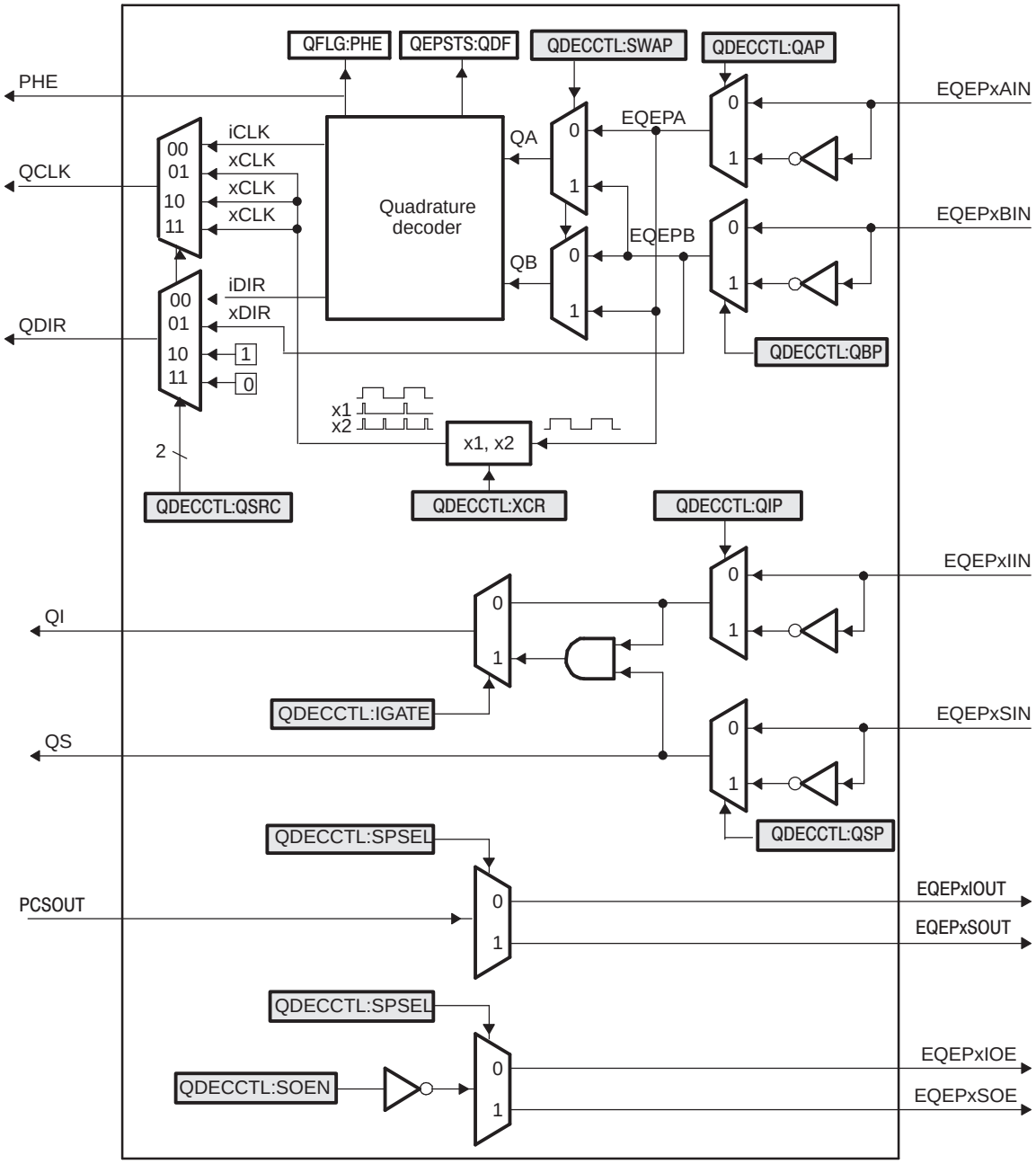
Table 15-1. EQEP Memory Map

Address Offset	Acronym	Register Description	Size(×16)/ #shadow	Reset
00h	QPOSCNT	eQEP Position Counter Register	2/0	0000 0000h
04h	QPOSINIT	eQEP Initialization Position Count Register	2/0	0000 0000h
08h	QPOSMAX	eQEP Maximum Position Count Register	2/0	0000 0000h
0Ch	QPOSCMP	eQEP Position-compare Register	2/1	0000 0000h
10h	QPOSILAT	eQEP Index Position Latch	2/0	0000 0000h
14h	QPOSSLAT	eQEP Strobe Position Latch	2/0	0000 0000h
18h	QPOSLAT	eQEP Position Latch	2/0	0000 0000h
1Ch	QUTMR	eQEP Unit Timer Register	2/0	0000 0000h
20h	QUPRD	eQEP Unit Period Register	2/0	0000 0000h
26h	QWDTMR	eQEP Watchdog Timer Register	1/0	0000h
24h	QWDPRD	eQEP Watchdog Period Register	1/0	0000h
2Ah	QDECCTL	eQEP Decoder Control Register	1/0	0000h
28h	QEPCTL	eQEP Control Register	1/0	0000h
2Eh	QCAPCTL	eQEP Capture Control Register	1/0	0000h
2Ch	QPOSCTL	eQEP Position-compare Control Register	1/0	0000h
32h	QEINT	eQEP Interrupt Enable Register	1/0	0000h
30h	QFLG	eQEP Interrupt Flag Register	1/0	0000h
36h	QCLR	eQEP Interrupt Clear Register	1/0	0000h
34h	QFRC	eQEP Interrupt Force Register	1/0	0000h
3Ah	QEPSTS	eQEP Status Register	1/0	0000h
38h	QCTMR	eQEP Capture Timer Register	1/0	0000h
3Eh	QCPRD	eQEP Capture Period Register	1/0	0000h
3Ch	QCTMRLAT	eQEP Capture Timer Latch	1/0	0000h
42h	QCPRDLAT	eQEP Capture Period Latch	1/0	0000h
40h	Reserved	Reserved	–	–

15.3 Quadrature Decoder Unit (QDU)

Figure 15-5 shows a functional block diagram of the QDU.

Figure 15-5. Functional Block Diagram of Decoder Unit



15.3.1 Position Counter Input Modes

Clock and direction input to position counter is selected using QDECCTL[QSRC] bits, based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- Up-count mode
- Down-count mode

15.3.1.1 Quadrature Count Mode

The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

Direction Decoding— The direction decoding logic of the eQEP circuit determines which one of the sequences (QEPA, QEPB) is the leading sequence and accordingly updates the direction information in QEPSTS[QDF] bit. Table 15-2 and Figure 15-6 show the direction decoding logic in truth table and state machine form. Both edges of the QEPA and QEPB signals are sensed to generate count pulses for the position counter. Therefore, the frequency of the clock generated by the eQEP logic is four times that of each input sequence. Figure 15-7 shows the direction decoding and clock generation from the eQEP input signals.

Table 15-2. Quadrature Decoder Truth Table

Previous Edge	Present Edge	QDIR	QPOSCNT
QA↑	QB↑	UP	Increment
	QB↓	DOWN	Decrement
	QA↓	TOGGLE	Increment or Decrement
QA↓	QB↓	UP	Increment
	QB↑	DOWN	Decrement
	QA↑	TOGGLE	Increment or Decrement
QB↑	QA↑	DOWN	Increment
	QA↓	UP	Decrement
	QB↓	TOGGLE	Increment or Decrement
QB↓	QA↓	DOWN	Increment
	QA↑	UP	Decrement
	QB↑	TOGGLE	Increment or Decrement

Figure 15-6. Quadrature Decoder State Machine

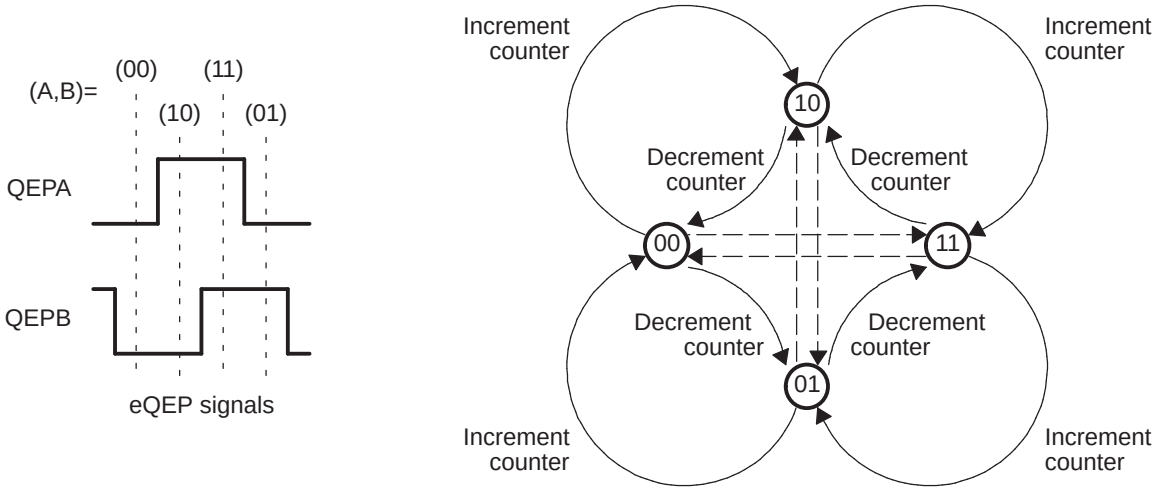
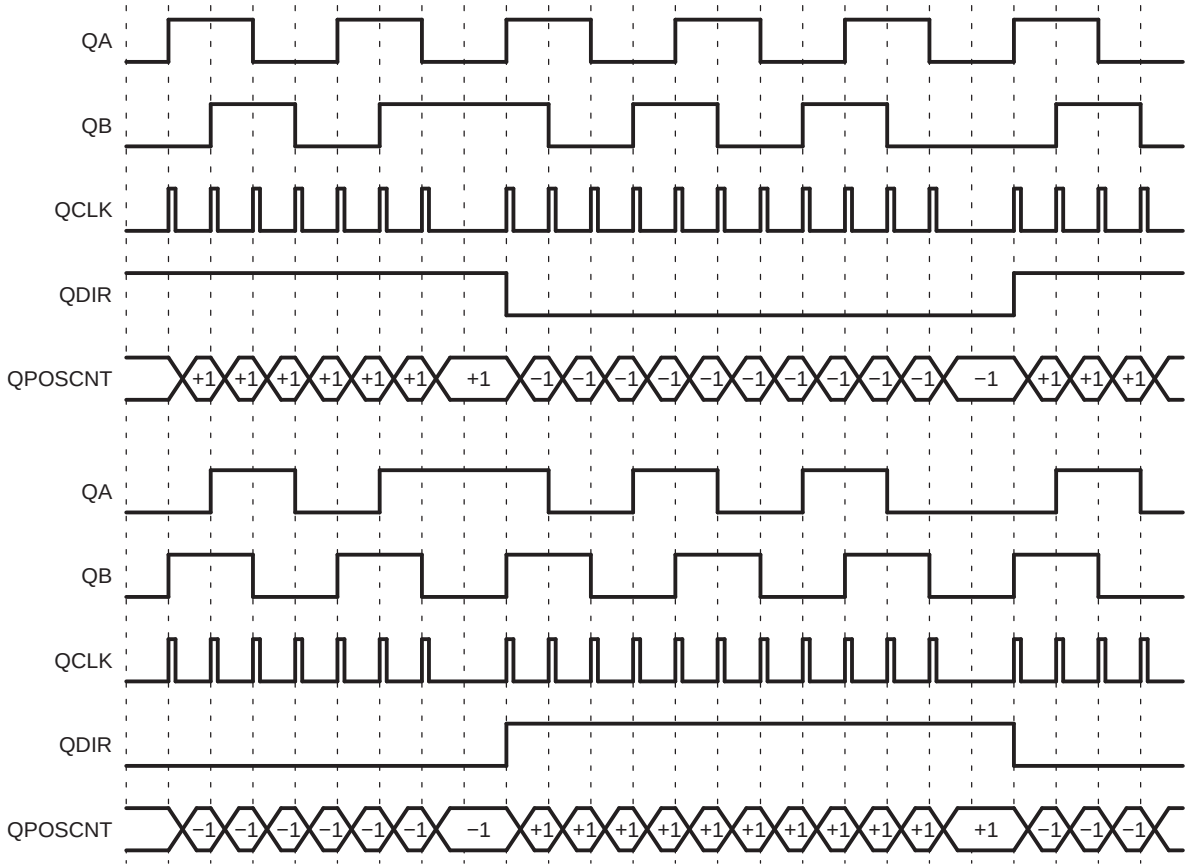


Figure 15-7. Quadrature-clock and Direction Decoding



Phase Error Flag— In normal operating conditions, quadrature inputs QEPA and QEPB will be 90 degrees out of phase. The phase error flag (PHE) is set in the QFLG register when edge transition is detected simultaneously on the QEPA and QEPB signals to optionally generate interrupts. State transitions marked by dashed lines in Figure 15-6 are invalid transitions that generate a phase error.

Count Multiplication— The eQEP position counter provides 4× the resolution of an input clock by generating a quadrature-clock (QCLK) on the rising/falling edges of both eQEP input clocks (QEPA and QEPB) as shown in Figure 15-7.

Reverse Count— In normal quadrature count operation, QEPA input is fed to the QA input of the quadrature decoder and the QEPB input is fed to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the QDECCTL register. This will swap the input to the quadrature decoder thereby reversing the counting direction.

15.3.1.2 Direction-count Mode

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. QEPA input will provide the clock for position counter and the QEPB input will have the direction information. The position counter is incremented on every rising edge of a QEPA input when the direction input is high and decremented when the direction input is low.

15.3.1.3 Up-Count Mode

The counter direction signal is hard-wired for up count and the position counter is used to measure the frequency of the QEPA input. Setting of the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of the QEPA input, thereby increasing the measurement resolution by 2× factor.

15.3.1.4 Down-Count Mode

The counter direction signal is hardwired for a down count and the position counter is used to measure the frequency of the QEPA input. Setting of the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of a QEPA input, thereby increasing the measurement resolution by 2× factor.

15.3.2 eQEP Input Polarity Selection

Each eQEP input can be inverted using QDECCTL[8:5] control bits. As an example, setting of QDECCTL[QIP] bit will invert the index input.

15.3.3 Position-Compare Sync Output

The enhanced eQEP peripheral includes a position-compare unit that is used to generate the position-compare sync signal on compare match between the position counter register (QPOSCNT) and the position-compare register (QPOSCMP). This sync signal can be output using an index pin or strobe pin of the EQEP peripheral.

Setting the QDECCTL[SOEN] bit enables the position-compare sync output and the QDECCTL[SPSEL] bit selects either an eQEP index pin or an eQEP strobe pin.

15.4 Position Counter and Control Unit (PCCU)

The position counter and control unit provides two configuration registers (QEPCTL and QPOSCTL) for setting up position counter operational modes, position counter initialization/latch modes and position-compare logic for sync signal generation.

15.4.1 Position Counter Operating Modes

Position counter data may be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse and position counter provides rotor angle with respect to index pulse position.

Position counter can be configured to operate in following four modes

- Position Counter Reset on Index Event
- Position Counter Reset on Maximum Position
- Position Counter Reset on the first Index Event
- Position Counter Reset on Unit Time Out Event (Frequency Measurement)

In all the above operating modes, position counter is reset to 0 on overflow and to QPOSMAX register value on underflow. Overflow occurs when the position counter counts up after QPOSMAX value. Underflow occurs when position counter counts down after 0. Interrupt flag is set to indicate overflow/underflow in QFLG register.

15.4.1.1 Position Counter Reset on Index Event (QEPCTL[PCRM] = 00)

If the index event occurs during the forward movement, then position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock.

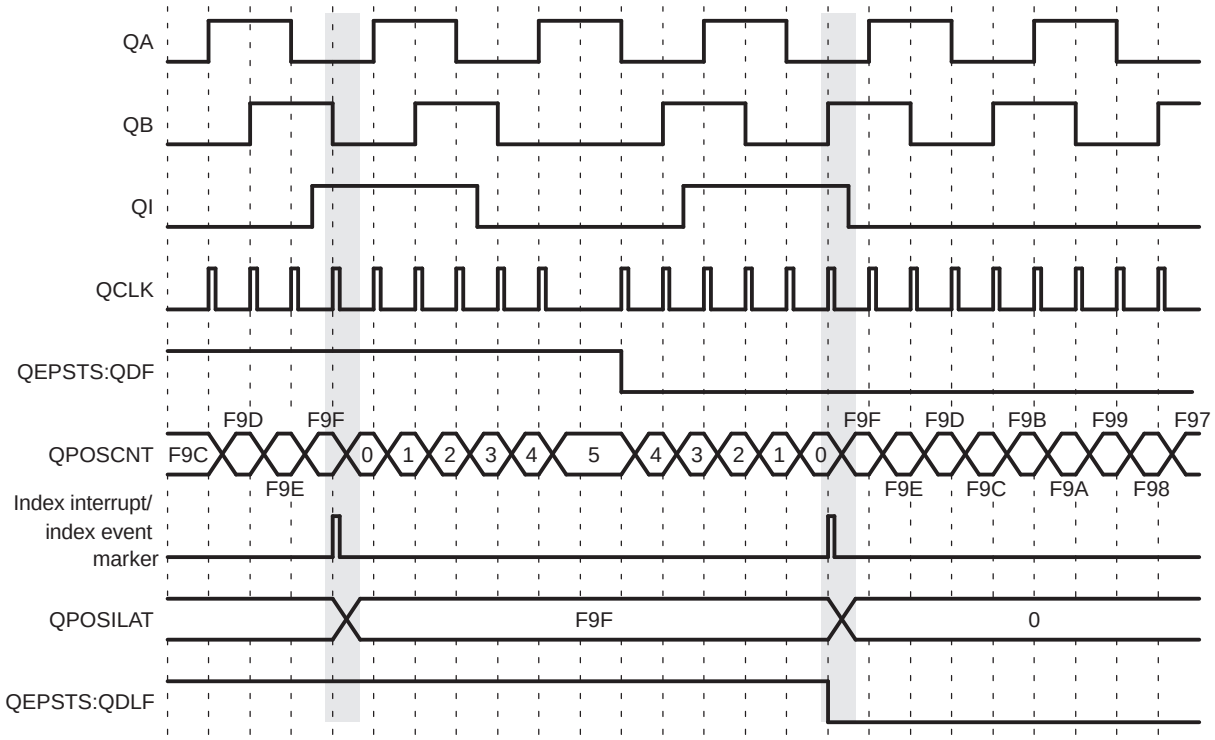
First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

For example, if the first reset operation occurs on the falling edge of QEPB during the forward direction, then all the subsequent reset must be aligned with the falling edge of QEPB for the forward rotation and on the rising edge of QEPB for the reverse rotation as shown in Figure 15-8.

The position-counter value is latched to the QPOSILAT register and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker. The position-counter error flag (QEPSTS[PCEF]) and error interrupt flag (QFLG[PCE]) are set if the latched value is not equal to 0 or QPOSMAX. The position-counter error flag (QEPSTS[PCEF]) is updated on every index event marker and an interrupt flag (QFLG[PCE]) will be set on error that can be cleared only through software.

The index event latch configuration QEPCTL[IEL] bits are ignored in this mode and position counter error flag/interrupt flag are generated only in index event reset mode.

Figure 15-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or F9Fh)

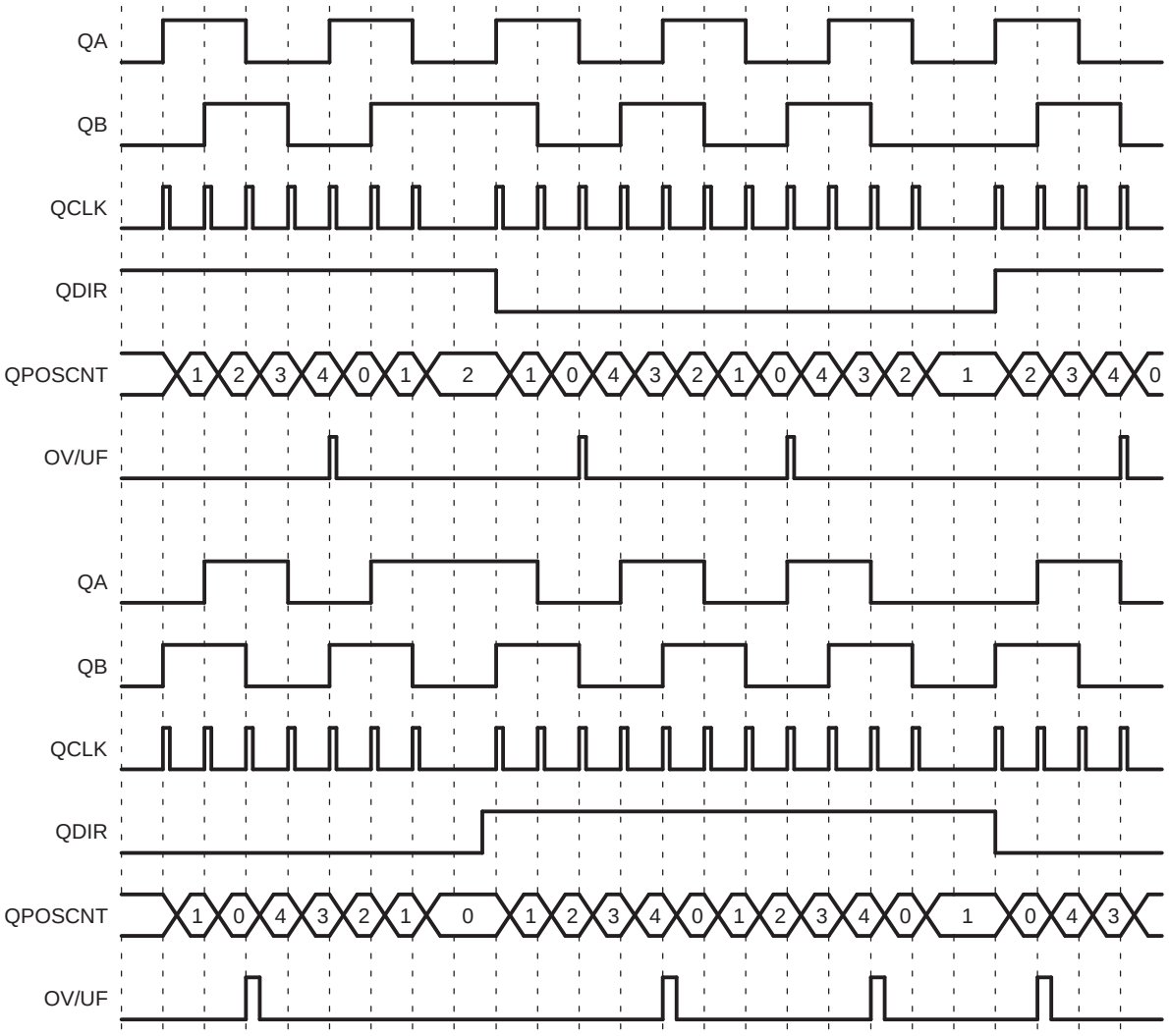


15.4.1.2 Position Counter Reset on Maximum Position (QEPCTL[PCRM] = 01)

If the position counter is equal to QPOSMAX, then the position counter is reset to 0 on the next eQEP clock for forward movement and position counter overflow flag is set. If the position counter is equal to ZERO, then the position counter is reset to QPOSMAX on the next QEP clock for reverse movement and position counter underflow flag is set. Figure 15-9 shows the position counter reset operation in this mode.

First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers; it also remembers the quadrature edge on the first index marker so that the same relative quadrature transition is used for the software index marker (QEPCTL[IEL]=11).

Figure 15-9. Position Counter Underflow/Overflow (QPOSMAX = 4)



15.4.1.3 Position Counter Reset on the First Index Event (QEPCTL[PCRM] = 10)

If the index event occurs during forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock. Note that this is done only on the first occurrence and subsequently the position counter value is not reset on an index event; rather, it is reset based on maximum position as described in Section 15.4.1.2.

First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for software index marker (QEPCTL[IEL]=11).

15.4.1.4 Position Counter Reset on Unit Time out Event (QEPCTL[PCRM] = 11)

In this mode, the QPOSCNT value is latched to the QPOSLAT register and then the QPOSCNT is reset (to 0 or QPOSMAX, depending on the direction mode selected by QDECCTL[QSRC] bits on a unit time event). This is useful for frequency measurement.

15.4.2 Position Counter Latch

The eQEP index and strobe input can be configured to latch the position counter (QPOSCNT) into QPOSILAT and QPOSSLAT, respectively, on occurrence of a definite event on these pins.

15.4.2.1 Index Event Latch

In some applications, it may not be desirable to reset the position counter on every index event and instead it may be required to operate the position counter in full 32-bit mode (QEPCTL[PCRM] = 01 and QEPCTL[PCRM] = 10 modes).

In such cases, the eQEP position counter can be configured to latch on the following events and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker.

- Latch on Rising edge (QEPCTL[IEL] = 01)
- Latch on Falling edge (QEPCTL[IEL] = 10)
- Latch on Index Event Marker (QEPCTL[IEL] = 11)

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

The index event latch interrupt flag (QFLG[IEL]) is set when the position counter is latched to the QPOSILAT register. The index event latch configuration bits (QEPCTZ[IEL]) are ignored when QEPCTL[PCRM] = 00.

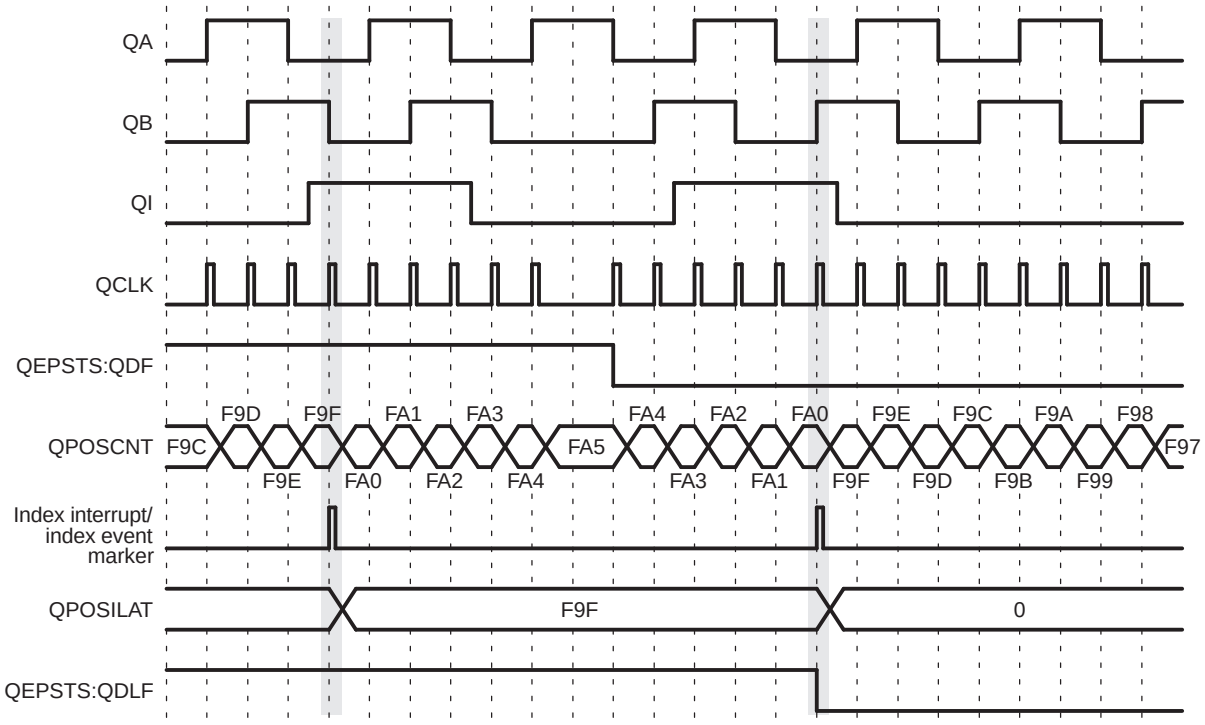
Latch on Rising Edge (QEPCTL[IEL] = 01)— The position counter value (QPOSCNT) is latched to the QPOSILAT register on every rising edge of an index input.

Latch on Falling Edge (QEPCTL[IEL] = 10)— The position counter value (QPOSCNT) is latched to the QPOSILAT register on every falling edge of index input.

Latch on Index Event Marker/Software Index Marker (QEPCTL[IEL] = 11— The first index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for latching the position counter (QEPCTL[IEL]=11).

Figure 15-10 shows the position counter latch using an index event marker.

Figure 15-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)



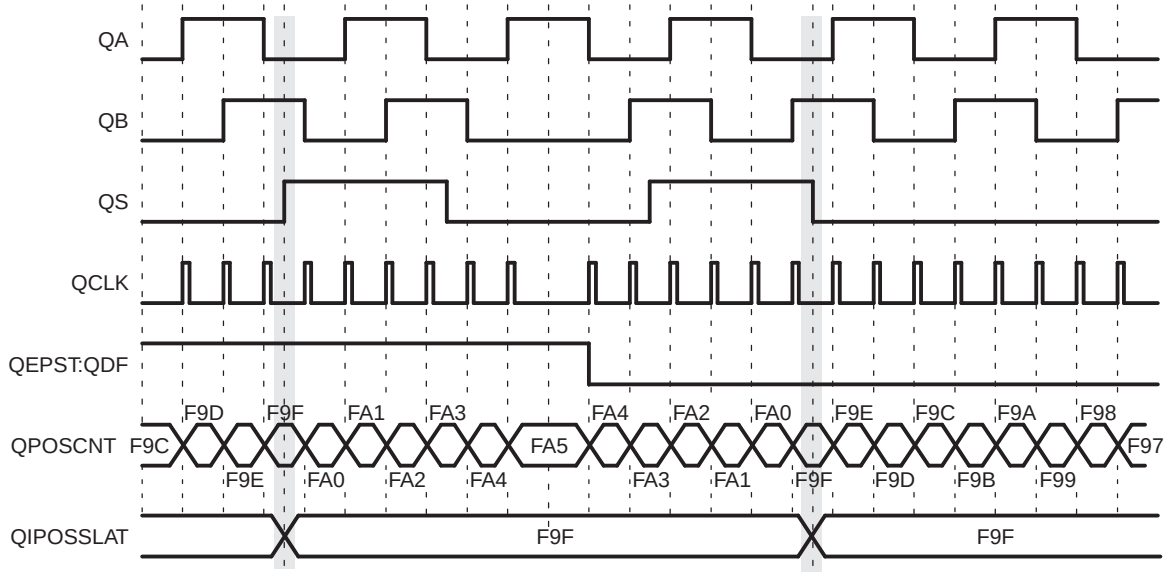
15.4.2.2 Strobe Event Latch

The position-counter value is latched to the QPOSSLAT register on the rising edge of the strobe input by clearing the QEPCTL[SEL] bit.

If the QEPCTL[SEL] bit is set, then the position counter value is latched to the QPOSSLAT register on the rising edge of the strobe input for forward direction and on the falling edge of the strobe input for reverse direction as shown in Figure 15-11.

The strobe event latch interrupt flag (QFLG[SEL) is set when the position counter is latched to the QPOSSLAT register.

Figure 15-11. Strobe Event Latch (QEPCTL[SEL] = 1)



15.4.3 Position Counter Initialization

The position counter can be initialized using following events:

- Index event
- Strobe event
- Software initialization

Index Event Initialization (IEI)— The QEPI index input can be used to trigger the initialization of the position counter at the rising or falling edge of the index input. If the QEPCTL[IEI] bits are 10, then the position counter (QPOSCNT) is initialized with a value in the QPOSINIT register on the rising edge of index input. Conversely, if the QEPCTL[IEI] bits are 11, initialization will be on the falling edge of the index input.

Strobe Event Initialization (SEI)— If the QEPCTL[SEI] bits are 10, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input.

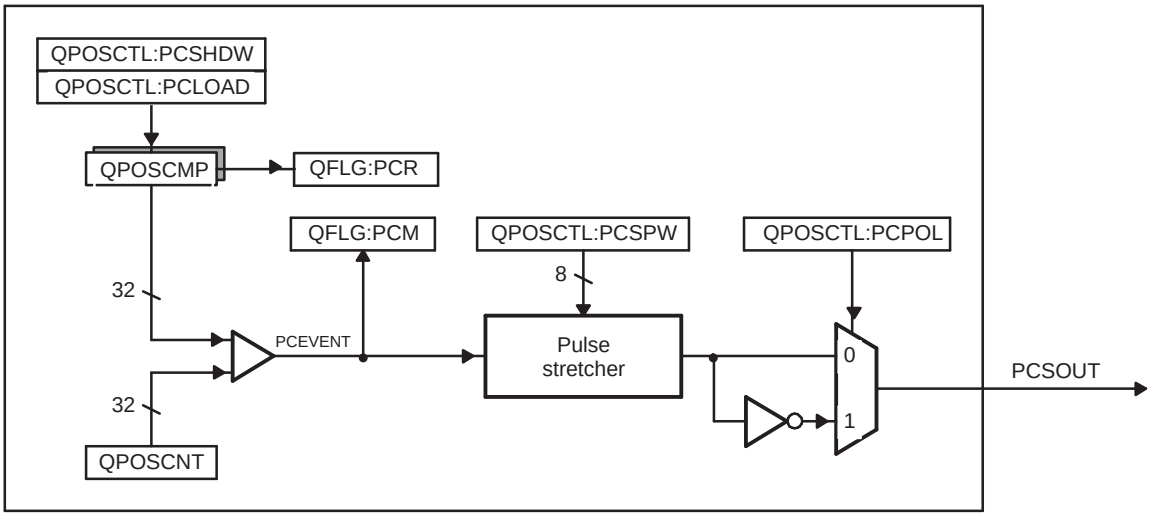
If QEPCTL[SEL] bits are 11, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

Software Initialization (SWI)— The position counter can be initialized in software by writing a 1 to the QEPCTL[SWI] bit. This bit is not automatically cleared. While the bit is still set, if a 1 is written to it again, the position counter will be re-initialized.

15.4.4 eQEP Position-compare Unit

The eQEP peripheral includes a position-compare unit that is used to generate a sync output and/or interrupt on a position-compare match. Figure 15-12 shows a diagram. The position-compare (QPOSCMP) register is shadowed and shadow mode can be enabled or disabled using the QPOSCTL[PSSHDW] bit. If the shadow mode is not enabled, the CPU writes directly to the active position compare register.

Figure 15-12. eQEP Position-compare Unit



In shadow mode, you can configure the position-compare unit (QPOSCTL[PCLOAD]) to load the shadow register value into the active register on the following events and to generate the position-compare ready (QFLG[PCR]) interrupt after loading.

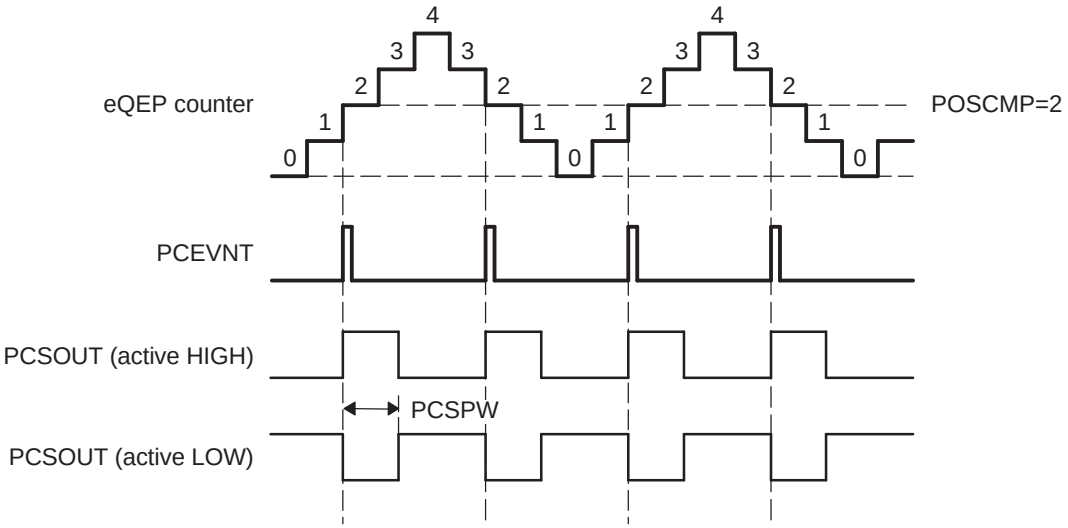
- Load on compare match
- Load on position-counter zero event

The position-compare match (QFLG[PCM]) is set when the position-counter value (QPOSCNT) matches with the active position-compare register (QPOSCMP) and the position-compare sync output of the programmable pulse width is generated on compare match to trigger an external device.

For example, if QPOSCMP = 2, the position-compare unit generates a position-compare event on 1 to 2 transitions of the eQEP position counter for forward counting direction and on 3 to 2 transitions of the eQEP position counter for reverse counting direction (see Figure 15-13).

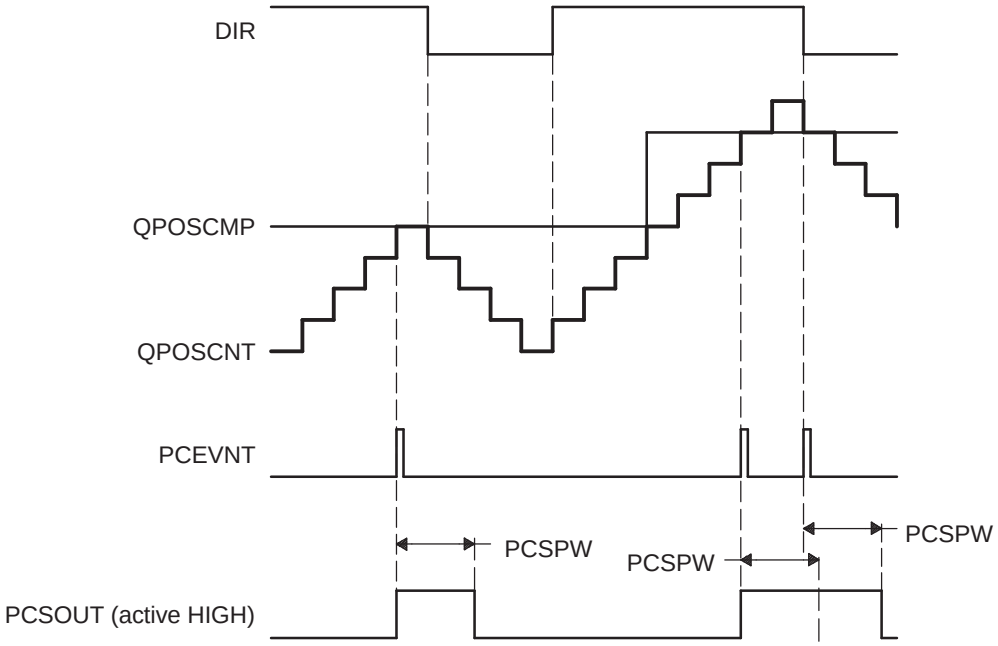
Section 15.9.15 shows the layout of the eQEP Position-Compare Control Register (QPOSCTL) and describes the QPOSCTL bit fields.

Figure 15-13. eQEP Position-compare Event Generation Points



The pulse stretcher logic in the position-compare unit generates a programmable position-compare sync pulse output on the position-compare match. In the event of a new position-compare match while a previous position-compare pulse is still active, then the pulse stretcher generates a pulse of specified duration from the new position-compare event as shown in Figure 15-14.

Figure 15-14. eQEP Position-compare Sync Output Pulse Stretcher



15.5 eQEP Edge Capture Unit

The eQEP peripheral includes an integrated edge capture unit to measure the elapsed time between the unit position events as shown in Figure 15-15. This feature is typically used for low-speed measurement using the following equation:

$$v(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (28)$$

where,

- X - Unit position is defined by integer multiple of quadrature edges (see Figure 15-16)
- ΔT - Elapsed time between unit position events
- v(k) - Velocity at time instant "k"

The eQEP capture timer (QCTMR) runs from prescaled VCLK and the prescaler is programmed by the QCAPCTL[CCPS] bits. The capture timer (QCTMR) value is latched into the capture period register (QCPRD) on every unit position event and then the capture timer is reset, a flag is set in QEPSTS:UPEVNT to indicate that new value is latched into the QCPRD register. Software can check this status flag before reading the period register for low speed measurement and clear the flag by writing 1.

Time measurement (ΔT) between unit position events will be correct if the following conditions are met:

- No more than 65,535 counts have occurred between unit position events.
- No direction change between unit position events.

The capture unit sets the eQEP overflow error flag (QEPSTS[COEF]) in the event of capture timer overflow between unit position events. If a direction change occurs between the unit position events, then an error flag is set in the status register (QEPSTS[CDEF]).

Capture Timer (QCTMR) and Capture period register (QCPRD) can be configured to latch on following events.

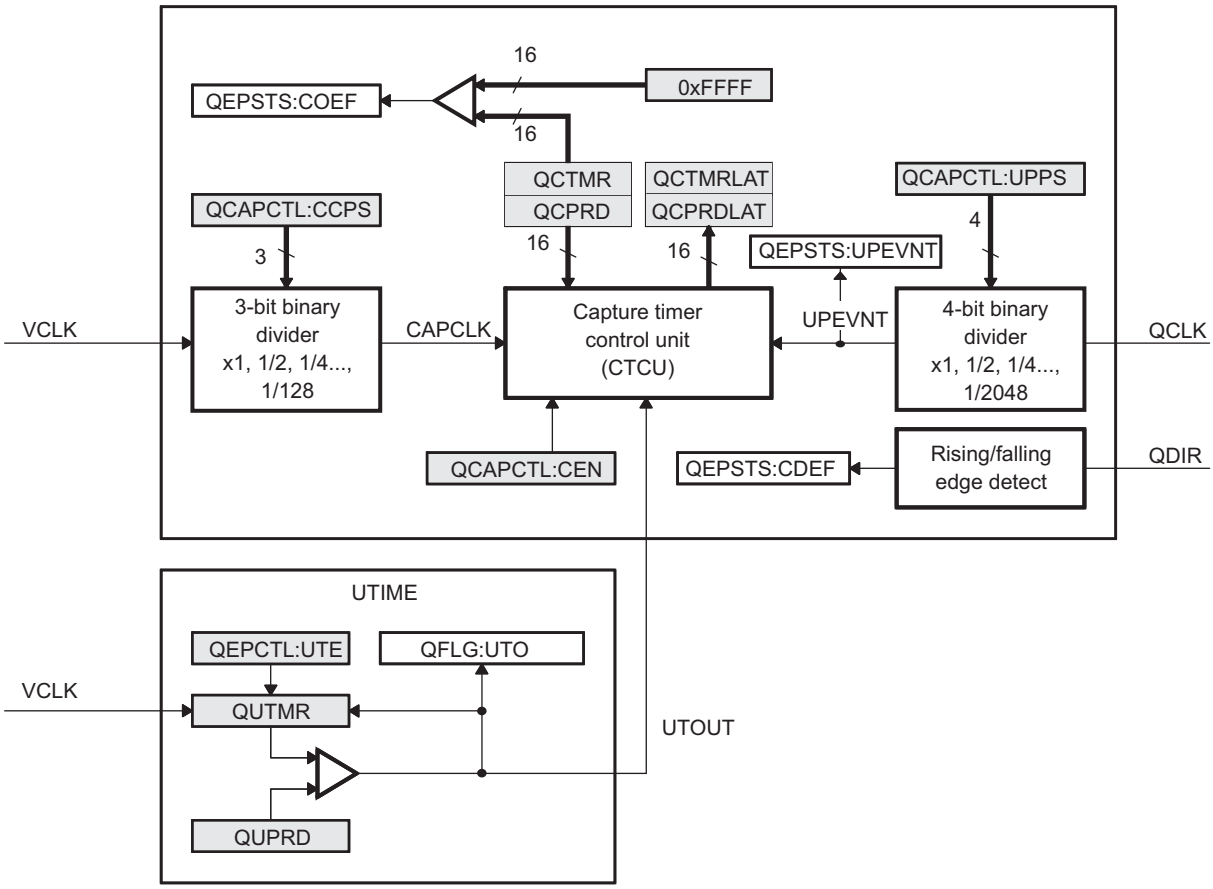
- CPU read of QPOSCNT register
- Unit time-out event

If the QEPCTL[QCLM] bit is cleared, then the capture timer and capture period values are latched into the QCTMRLAT and QCPRDLAT registers, respectively, when the CPU reads the position counter (QPOSCNT).

If the QEPCTL[QCLM] bit is set, then the position counter, capture timer, and capture period values are latched into the QPOSLAT, QCTMRLAT and QCPRDLAT registers, respectively, on unit time out.

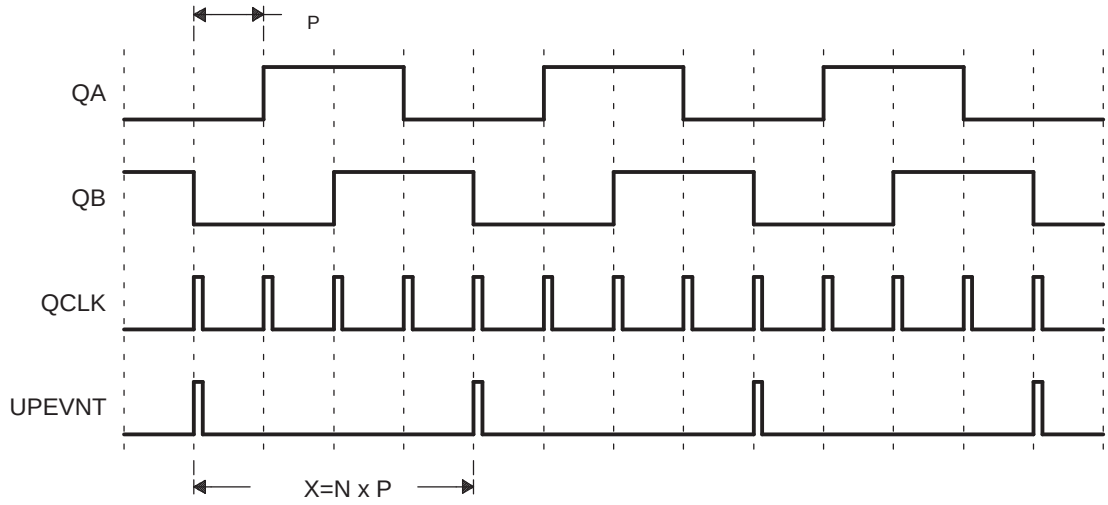
Figure 15-17 shows the capture unit operation along with the position counter.

Figure 15-15. eQEP Edge Capture Unit



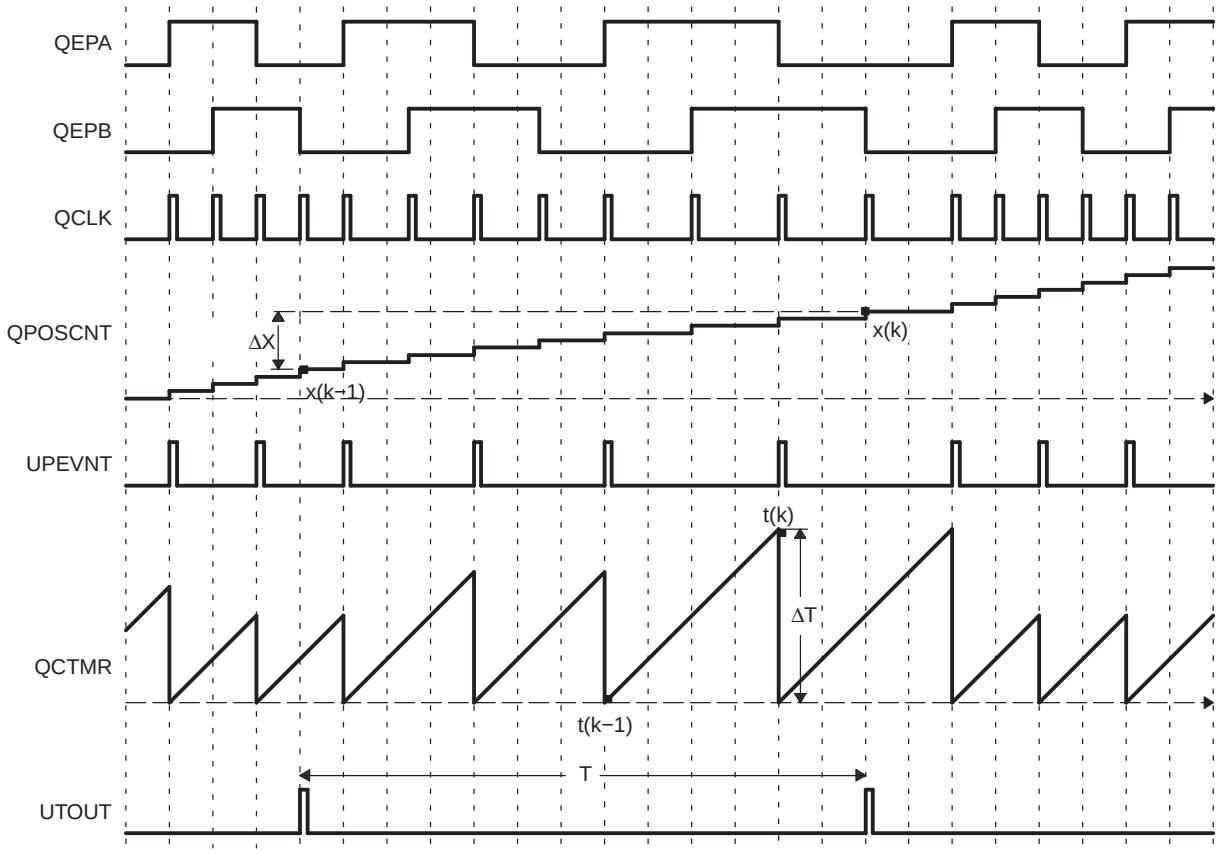
NOTE: The QCAPCTL[UPPS] prescaler should not be modified dynamically (such as switching the unit event prescaler from QCLK/4 to QCLK/8). Doing so may result in undefined behavior. The QCAPCTL[CPPS] prescaler can be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLK/4 to SYSCLK/8) only after the capture unit is disabled.

Figure 15-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)



N - Number of quadrature periods selected using QCAPCTL[UPPS] bits

Figure 15-17. eQEP Edge Capture Unit - Timing Details



Velocity Calculation Equations:

$$v(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (29)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement

t(k): Time instant "k"

t(k-1): Time instant "k-1"

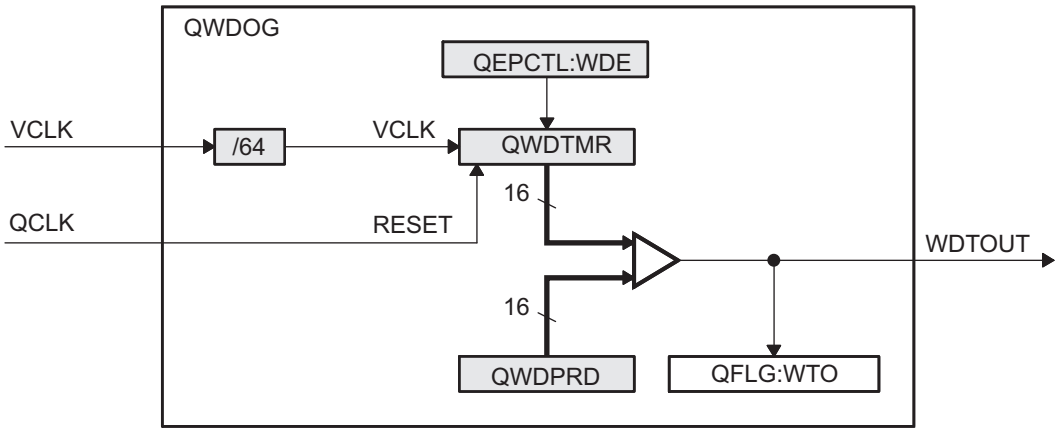
Unit time (T) and unit period(X) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QPOSLAT and QCPRDLAT registers.

Parameter	Relevant Register to Configure or Read the Information
T	Unit Period Register (QUPRD)
ΔX	Incremental Position = QPOSLAT(k) - QPOSLAT(K-1)
X	Fixed unit position defined by sensor resolution and ZCAPCTL[UPPS] bits
ΔT	Capture Period Latch (QCPRDLAT)

15.6 eQEP Watchdog

The eQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper operation of the motion-control system. The eQEP watchdog timer is clocked from VCLK/64 and the quadrate clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match (QWDPRD = QWDTMR), then the watchdog timer will time out and the watchdog interrupt flag will be set (QFLG[WTO]). The time-out value is programmable through the watchdog period register (QWDPRD).

Figure 15-18. eQEP Watchdog Timer

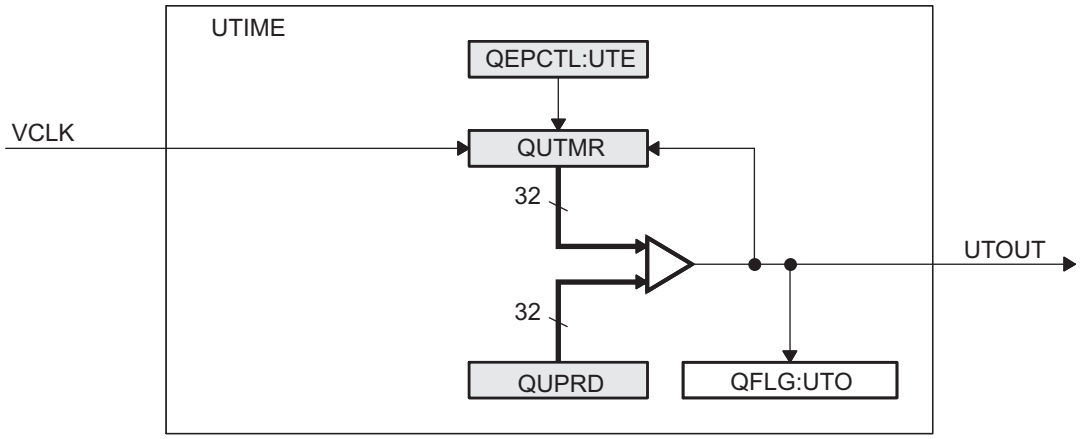


15.7 Unit Timer Base

The eQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by VCLK to generate periodic interrupts for velocity calculations. The unit time out interrupt is set (QFLG[UTO]) when the unit timer (QUTMR) matches the unit period register (QUPRD).

The eQEP peripheral can be configured to latch the position counter, capture timer, and capture period values on a unit time out event so that latched values are used for velocity calculation as described in Section 15.5.

Figure 15-19. eQEP Unit Time Base

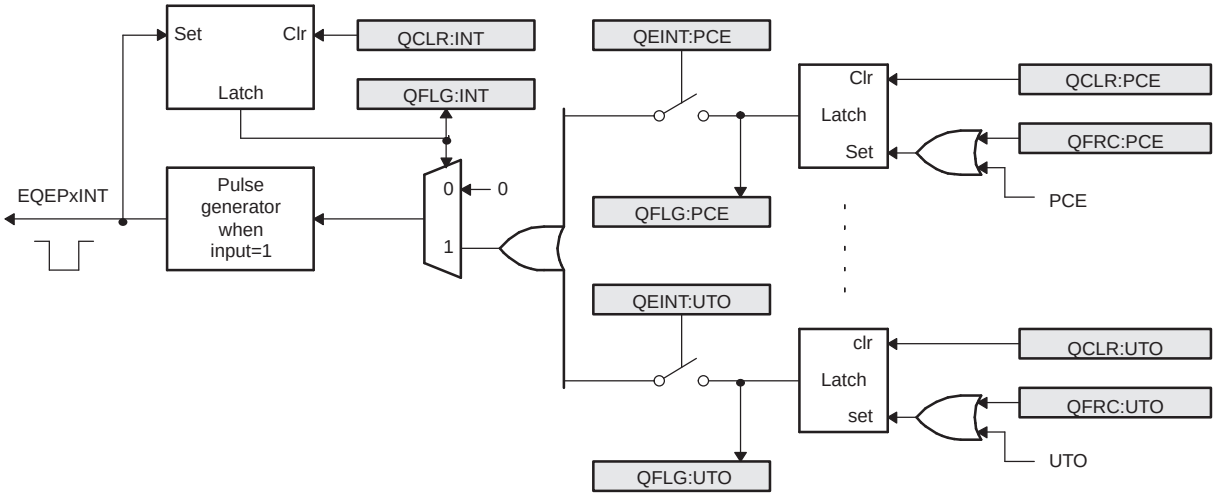


15.8 eQEP Interrupt Structure

Figure 15-20 shows how the interrupt mechanism works in the EQEP module.

Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL and UTO) can be generated. The interrupt control register (QEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (QFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated only to the PIE if any of the interrupt events is enabled, the flag bit is 1 and the INT flag bit is 0. The interrupt service routine will need to clear the global interrupt flag bit and the serviced event, via the interrupt clear register (QCLR), before any other interrupt pulses are generated. You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

Figure 15-20. EQEP Interrupt Generation



15.9 eQEP Registers

Table 15-3 lists the eQEP registers. The base address for the control registers is FCF7 9900h.

Table 15-3. eQEP Registers

Offset	Acronym	Register Description	Section
00h	QPOSCNT	eQEP Position Counter Register	Section 15.9.1
04h	QPOSINIT	eQEP Initialization Position Count Register	Section 15.9.2
08h	QPOSMAX	eQEP Maximum Position Count Register	Section 15.9.3
0Ch	QPOSCMP	eQEP Position-compare Register	Section 15.9.4
10h	QPOSILAT	eQEP Index Position Latch Register	Section 15.9.5
14h	QPOSSLAT	eQEP Strobe Position Latch Register	Section 15.9.6
18h	QPOSLAT	eQEP Position Latch Register	Section 15.9.7
1Ch	QUTMR	eQEP Unit Timer Register	Section 15.9.8
20h	QUPRD	eQEP Unit Period Register	Section 15.9.9
26h	QWDTMR	eQEP Watchdog Timer Register	Section 15.9.10
24h	QWDPRD	eQEP Watchdog Period Register	Section 15.9.11
2Ah	QDECCTL	eQEP Decoder Control Register	Section 15.9.12
28h	QEPCTL	eQEP Control Register	Section 15.9.13
2Eh	QCAPCTL	eQEP Capture Control Register	Section 15.9.14
2Ch	QPOSCTL	eQEP Position-Compare Control Register	Section 15.9.15
32h	QEINT	eQEP Interrupt Enable Register	Section 15.9.16
30h	QFLG	eQEP Interrupt Flag Register	Section 15.9.17
36h	QCLR	eQEP Interrupt Clear Register	Section 15.9.18
34h	QFRC	eQEP Interrupt Force Register	Section 15.9.19
3Ah	QEPSTS	eQEP Status Register	Section 15.9.20
38h	QCTMR	eQEP Capture Timer Register	Section 15.9.21
3Eh	QCPRD	eQEP Capture Period Register	Section 15.9.22
3Ch	QCTMRLAT	eQEP Capture Timer Latch Register	Section 15.9.23
42h	QCPRDLAT	eQEP Capture Period Latch Register	Section 15.9.24

15.9.1 eQEP Position Counter Register (QPOSCNT)

Figure 15-21. eQEP Position Counter (QPOSCNT) Register (offset = 00h)

31	0
QPOSCNT	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-4. eQEP Position Counter (QPOSCNT) Register Field Descriptions

Bit	Field	Description
31-0	QPOSCNT	This 32-bit position counter register counts up/down on every eQEP pulse based on direction input. This counter acts as a position integrator whose count value is proportional to position from a give reference point.

15.9.2 eQEP Position Counter Initialization Register (QPOSINIT)

Figure 15-22. eQEP Position Counter Initialization (QPOSINIT) Register (offset = 04h)

31	0
QPOSINIT	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-5. eQEP Position Counter Initialization (QPOSINIT) Register Field Descriptions

Bit	Field	Description
31-0	QPOSINIT	This register contains the position value that is used to initialize the position counter based on external strobe or index event. The position counter can be initialized through software.

15.9.3 eQEP Maximum Position Count Register (QPOSMAX)

Figure 15-23. eQEP Maximum Position Count Register (QPOSMAX) Register (offset = 08h)

31	0
QPOSMAX	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-6. eQEP Maximum Position Count (QPOSMAX) Register Field Descriptions

Bit	Field	Description
31-0	QPOSMAX	This register contains the maximum position counter value.

15.9.4 eQEP Position-Compare Register (QPOSCMP)

Figure 15-24. eQEP Position-Compare (QPOSCMP) Register (offset = 0Ch)

31	0
QPOSCMP	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-7. eQEP Position-Compare (QPOSCMP) Register Field Descriptions

Bit	Field	Description
31-0	QPOSCMP	The position-compare value in this register is compared with the position counter (QPOSCNT) to generate sync output and/or interrupt on compare match.

15.9.5 eQEP Index Position Latch Register (QPOSILAT)

Figure 15-25. eQEP Index Position Latch (QPOSILAT) Register (offset = 10h)

31	0
QPOSILAT	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 15-8. eQEP Index Position Latch (QPOSILAT) Register Field Descriptions

Bit	Field	Description
31-0	QPOSILAT	The position-counter value is latched into this register on an index event as defined by the QEPCTL[IEL] bits.

15.9.6 eQEP Strobe Position Latch Register (QPOSSLAT)

Figure 15-26. eQEP Strobe Position Latch (QPOSSLAT) Register (offset = 14h)

31	0
QPOSSLAT	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 15-9. eQEP Strobe Position Latch (QPOSSLAT) Register Field Descriptions

Bit	Field	Description
31-0	QPOSSLAT	The position-counter value is latched into this register on strobe event as defined by the QEPCTL[SEL] bits.

15.9.7 eQEP Position Counter Latch Register (QPOSLAT)

Figure 15-27. eQEP Position Counter Latch (QPOSLAT) Register (offset = 18h)

31	0
QPOSLAT	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 15-10. eQEP Position Counter Latch (QPOSLAT) Register Field Descriptions

Bit	Field	Description
31-0	QPOSLAT	The position-counter value is latched into this register on unit time out event.

15.9.8 eQEP Unit Timer Register (QUTMR)

Figure 15-28. eQEP Unit Timer (QUTMR) Register (offset = 1Ch)

31	0
QUTMR	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-11. eQEP Unit Timer (QUTMR) Register Field Descriptions

Bit	Field	Description
31-0	QUTMR	This register acts as time base for unit time event generation. When this timer value matches with unit time period value, unit time event is generated.

15.9.9 eQEP Register Unit Period Register (QUPRD)

Figure 15-29. eQEP Register Unit Period (QUPRD) Register (offset = 20h)

31	0
QUPRD	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-12. eQEP Unit Period (QUPRD) Register Field Descriptions

Bit	Field	Description
31-0	QUPRD	This register contains the period count for unit timer to generate periodic unit time events to latch the eQEP position information at periodic interval and optionally to generate interrupt.

15.9.10 eQEP Watchdog Timer Register (QWDTMR)

Figure 15-30. eQEP Watchdog Timer (QWDTMR) Register (offset = 26h)

15	0
QWDTMR	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-13. eQEP Watchdog Timer (QWDTMR) Register Field Descriptions

Bit	Field	Description
15-0	QWDTMR	This register acts as time base for watch dog to detect motor stalls. When this timer value matches with watch dog period value, watch dog timeout interrupt is generated. This register is reset upon edge transition in quadrature-clock indicating the motion.

15.9.11 eQEP Watchdog Period Register (QWDPRD)

Figure 15-31. eQEP Watchdog Period (QWDPRD) Register (offset = 24h)

15	0
QWDPRD	
R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 15-14. eQEP Watchdog Period (QWDPRD) Register Field Description

Bit	Field	Description
15-0	QWDPRD	This register contains the time-out count for the eQEP peripheral watch dog timer. When the watchdog timer value matches the watchdog period value, a watchdog timeout interrupt is generated.

15.9.12 eQEP Decoder Control Register (QDECCTL)

Figure 15-32. eQEP Decoder Control (QDECCTL) Register (offset = 2Ah)

15	14	13	12	11	10	9	8
QSRC		SOEN	SPSEL	XCR	SWAP	IGATE	QAP
R/W-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	0
QBP	QIP	QSP	Reserved	
R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-15. eQEP Decoder Control (QDECCTL) Register Field Descriptions

Bit	Field	Value	Description
15-14	QSRC		Position-counter source selection.
		0	Quadrature count mode (QCLK = iCLK, QDIR = iDIR)
		1h	Direction-count mode (QCLK = xCLK, QDIR = xDIR)
		2h	UP count mode for frequency measurement (QCLK = xCLK, QDIR = 1)
		3h	DOWN count mode for frequency measurement (QCLK = xCLK, QDIR = 0)
13	SOEN		Sync output-enable.
		0	Disable position-compare sync output.
		1	Enable position-compare sync output.
12	SPSEL		Sync output pin selection.
		0	Index pin is used for sync output.
		1	Strobe pin is used for sync output.
11	XCR		External clock rate.
		0	2x resolution: Count the rising/falling edge.
		1	1x resolution: Count the rising edge only.
10	SWAP		Swap quadrature clock inputs. This swaps the input to the quadrature decoder, reversing the counting direction.
		0	Quadrature-clock inputs are not swapped.
		1	Quadrature-clock inputs are swapped.
9	IGATE		Index pulse gating option.
		0	Disable gating of Index pulse.
		1	Gate the index pin with strobe.
8	QAP		QEPA input polarity.
		0	No effect.
		1	Negates QEPA input.
7	QBP		QEPB input polarity.
		0	No effect.
		1	Negates QEPB input.
6	QIP		QEPI input polarity.
		0	No effect.
		1	Negates QEPI input.
5	QSP		QEPS input polarity.
		0	No effect.
		1	Negates QEPS input.
4-0	Reserved	0	Always write as 0.

15.9.13 eQEP Control Register (QEPCTL)

Figure 15-33. eQEP Control (QEPCTL) Register (offset = 28h)

15	14	13	12	11	10	9	8
FREE, SOFT		PCRM		SEI		IEI	
R/W-0		R/W-0		R/W-0		R/W-0	

7	6	5	4	3	2	1	0
SWI	SEL	IEL		QPEN	QCLM	UTE	WDE
R/W-0	R/W-0	R/W-0		R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 15-16. eQEP Control (QEPCTL) Register Field Descriptions

Bit	Field	Value	Description
15-14	FREE, SOFT		Emulation Control Bits.
			QPOSCNT behavior:
		0	Position counter stops immediately on emulation suspend.
		1h	Position counter continues to count until the rollover.
		2h-3h	Position counter is unaffected by emulation suspend.
			QWDTMR behavior:
		0	Watchdog counter stops immediately.
		1h	Watchdog counter counts until WD period match roll over.
		2h-3h	Watchdog counter is unaffected by emulation suspend.
			QUTMR behavior:
		0	Unit timer stops immediately.
		1h	Unit timer counts until period rollover.
		2h-3h	Unit timer is unaffected by emulation suspend.
			QCTMR behavior:
		0	Capture Timer stops immediately.
		1h	Capture Timer counts until next unit period event.
		2h-3h	Capture Timer is unaffected by emulation suspend.
13-12	PCRM		Position counter reset mode.
		0	Position counter reset on an index event.
		1h	Position counter reset on the maximum position.
		2h	Position counter reset on the first index event.
		3h	Position counter reset on a unit time event.
11-10	SEI		Strobe event initialization of position counter.
		0	Does nothing (action is disabled).
		1h	Does nothing (action is disabled).
		2h	Initializes the position counter on rising edge of the QEPS signal.
		3h	Clockwise Direction: Initializes the position counter on the rising edge of QEPS strobe.
			Counter Clockwise Direction: Initializes the position counter on the falling edge of QEPS strobe.
9-8	IEI		Index event initialization of position counter.
		0	Do nothing (action is disabled).
		1h	Do nothing (action is disabled).
		2h	Initializes the position counter on the rising edge of the QEPI signal (QPOSCNT = QPOSINIT).
		3h	Initializes the position counter on the falling edge of QEPI signal (QPOSCNT = QPOSINIT).
7	SWI		Software initialization of position counter.
		0	Do nothing (action is disabled).
		1	Initialize position counter (QPOSCNT = QPOSINIT). This bit is not cleared automatically.

Table 15-16. eQEP Control (QEPCTL) Register Field Descriptions (continued)

Bit	Field	Value	Description
6	SEL		Strobe event latch of position counter.
		0	The position counter is latched on the rising edge of QEPS strobe (QPOSSLAT = POSCCNT). Latching on the falling edge can be done by inverting the strobe input using the QSP bit in the QDECCTL register.
		1	Clockwise Direction: Position counter is latched on rising edge of QEPS strobe.
			Counter Clockwise Direction: Position counter is latched on falling edge of QEPS strobe.
5-4	IEL		Index event latch of position counter (software index marker).
		0	Reserved
		1h	Latches position counter on rising edge of the index signal.
		2h	Latches position counter on falling edge of the index signal.
		3h	Software index marker. Latches the position counter and quadrature direction flag on index event marker. The position counter is latched to the QPOSILAT register and the direction flag is latched in the QEPSTS[QDLF] bit. This mode is useful for software index marking.
3	QPEN		Quadrature position counter enable/software reset.
		0	Reset the eQEP peripheral internal operating flags/read-only registers. Control/configuration registers are not disturbed by a software reset.
		1	eQEP position counter is enabled.
2	QCLM		eQEP capture latch mode.
		0	Latch on position counter read by CPU. Capture timer and capture period values are latched into QCTMRLAT and QCPRDLAT registers when CPU reads the QPOSCNT register.
		1	Latch on unit time out. Position counter, capture timer and capture period values are latched into QPOSLAT, QCTMRLAT and QCPRDLAT registers on unit time out.
1	UTE		eQEP unit timer enable.
		0	Disable eQEP unit timer.
		1	Enable eQEP unit timer.
0	WDE		eQEP watchdog enable.
		0	Disable eQEP watchdog timer.
		1	Enable eQEP watchdog timer.

15.9.14 eQEP Capture Control Register (QCAPCTL)

Figure 15-34. eQEP Capture Control (QCAPCTL) Register (offset = 2Eh)

15	14	8
CEN	Reserved	
R/W-0	R-0	

7	6	4	3	0
Reserved	CCPS		UPPS	
R-0	R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-17. eQEP Capture Control (QCAPCTL) Register Field Descriptions

Bit	Field	Value	Description
15	CEN		Enable eQEP capture.
		0	eQEP capture unit is disabled.
		1	eQEP capture unit is enabled.
14-7	Reserved	0	Always write as 0.
6-4	CCPS		eQEP capture timer clock prescaler.
		0	CAPCLK = VCLK/1
		1h	CAPCLK = VCLK/2
		2h	CAPCLK = VCLK/4
		3h	CAPCLK = VCLK/8
		4h	CAPCLK = VCLK/16
		5h	CAPCLK = VCLK/32
		6h	CAPCLK = VCLK/64
		7h	CAPCLK = VCLK/128
3-0	UPPS		Unit position event prescaler.
		0	UPEVNT = QCLK/1
		1h	UPEVNT = QCLK/2
		2h	UPEVNT = QCLK/4
		3h	UPEVNT = QCLK/8
		4h	UPEVNT = QCLK/16
		5h	UPEVNT = QCLK/32
		6h	UPEVNT = QCLK/64
		7h	UPEVNT = QCLK/128
		8h	UPEVNT = QCLK/256
		9h	UPEVNT = QCLK/512
		Ah	UPEVNT = QCLK/1024
		Bh	UPEVNT = QCLK/2048
		Ch-Fh	Reserved

15.9.15 eQEP Position-Compare Control Register (QPOSCTL)

Figure 15-35. eQEP Position-compare Control (QPOSCTL) Register (offset = 2Ch)

15	14	13	12	11	8
PCSHDW	PCLOAD	PCPOL	PCE	PCSPW	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

7	0
PCSPW	
R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 15-18. eQEP Position-compare Control (QPOSCTL) Register Field Descriptions

Bit	Field	Value	Description
15	PCSHDW		Position-compare shadow enable.
		0	Shadow disabled, load Immediate.
		1	Shadow enabled.
14	PCLOAD		Position-compare shadow load mode.
		0	Load on QPOSCNT = 0.
		1	Load when QPOSCNT = QPOSCMP.
13	PCPOL		Polarity of sync output.
		0	Active HIGH pulse output.
		1	Active LOW pulse output.
12	PCE		Position-compare enable.
		0	Disable position compare unit.
		1	Enable position compare unit.
11-0	PCSPW		Select-position-compare sync output pulse width.
		0	1 × 4 × VCLK cycles
		1h	2 × 4 × VCLK cycles
		:	:
		FFFh	4096 × 4 × VCLK cycles

15.9.16 eQEP Interrupt Enable Register (QEINT)

Figure 15-36. eQEP Interrupt Enable (QEINT) Register (offset = 32h)

15	12	11	10	9	8
Reserved		UTO	IEL	SEL	PCM
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	QPE	PCE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-19. eQEP Interrupt Enable (QEINT) Register Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Always write as 0.
11	UTO		Unit time out interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
10	IEL		Index event latch interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
9	SEL		Strobe event latch interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
8	PCM		Position-compare match interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
7	PCR		Position-compare ready interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
6	PCO		Position counter overflow interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
5	PCU		Position counter underflow interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
4	WTO		Watchdog time out interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
3	QDC		Quadrature direction change interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
2	QPE		Quadrature phase error interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
1	PCE		Position counter error interrupt enable.
		0	Interrupt is disabled.
		1	Interrupt is enabled.
0	Reserved	0	Reserved

15.9.17 eQEP Interrupt Flag Register (QFLG)

Figure 15-37. eQEP Interrupt Flag (QFLG) Register (offset = 30h)

15	12	11	10	9	8
Reserved		UTO	IEL	SEL	PCM
R-0		R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -*n* = value after reset

Table 15-20. eQEP Interrupt Flag (QFLG) Register Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Always write as 0.
11	UTO		Unit time out interrupt flag.
		0	No interrupt is generated.
		1	Set by eQEP unit timer period match.
10	IEL		Index event latch interrupt flag.
		0	No interrupt is generated.
		1	Set after latching the QPOSCNT to QPOSILAT.
9	SEL		Strobe event latch interrupt flag.
		0	No interrupt is generated.
		1	Set after latching the QPOSCNT to QPOSSLAT.
8	PCM		Position-compare match interrupt flag.
		0	No interrupt is generated.
		1	Set on position-compare match.
7	PCR		Position-compare ready interrupt flag.
		0	No interrupt is generated.
		1	Set after transferring the shadow register value to the active position compare register.
6	PCO		Position counter overflow interrupt flag.
		0	No interrupt is generated.
		1	Set on position counter overflow.
5	PCU		Position counter underflow interrupt flag.
		0	No interrupt is generated.
		1	Set on position counter underflow.
4	WTO		Watchdog time out interrupt flag.
		0	No interrupt is generated.
		1	Set by watchdog timeout.
3	QDC		Quadrature direction change interrupt flag.
		0	No interrupt is generated.
		1	Set during change of direction.
2	PHE		Quadrature phase error interrupt flag.
		0	No interrupt is generated.
		1	Set on simultaneous transition of QEPA and QEPB.
1	PCE		Position counter error interrupt flag.
		0	No interrupt is generated.
		1	Set on position counter error.
0	INT		Global interrupt status flag.
		0	No interrupt is generated.
		1	Interrupt is generated.

15.9.18 eQEP Interrupt Clear Register (QCLR)

Figure 15-38. eQEP Interrupt Clear (QCLR) Register (offset = 36h)

15	12	11	10	9	8
Reserved		UTO	IEL	SEL	PCM
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-21. eQEP Interrupt Clear (QCLR) Register Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Always write as 0.
11	UTO		Clear unit time out interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
10	IEL		Clear index event latch interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
9	SEL		Clear strobe event latch interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
8	PCM		Clear eQEP compare match event interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
7	PCR		Clear position-compare ready interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
6	PCO		Clear position counter overflow interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
5	PCU		Clear position counter underflow interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
4	WTO		Clear watchdog timeout interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
3	QDC		Clear quadrature direction change interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
2	PHE		Clear quadrature phase error interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
1	PCE		Clear position counter error interrupt flag.
		0	No effect.
		1	Clears the interrupt flag.
0	INT		Global interrupt clear flag.
		0	No effect.
		1	Clears the interrupt flag and enables further interrupts to be generated if an event flags is set to 1.

15.9.19 eQEP Interrupt Force Register (QFRC)

Figure 15-39. eQEP Interrupt Force (QFRC) Register (offset = 34h)

15	12	11	10	9	8
Reserved		UTO	IEL	SEL	PCM
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-22. eQEP Interrupt Force (QFRC) Register Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Always write as 0.
11	UTO		Force unit time out interrupt.
		0	No effect.
		1	Force the interrupt.
10	IEL		Force index event latch interrupt.
		0	No effect.
		1	Force the interrupt.
9	SEL		Force strobe event latch interrupt.
		0	No effect.
		1	Force the interrupt.
8	PCM		Force position-compare match interrupt .
		0	No effect.
		1	Force the interrupt.
7	PCR		Force position-compare ready interrupt.
		0	No effect.
		1	Force the interrupt.
6	PCO		Force position counter overflow interrupt.
		0	No effect.
		1	Force the interrupt.
5	PCU		Force position counter underflow interrupt.
		0	No effect.
		1	Force the interrupt.
4	WTO		Force watchdog time out interrupt.
		0	No effect.
		1	Force the interrupt.
3	QDC		Force quadrature direction change interrupt.
		0	No effect.
		1	Force the interrupt.
2	PHE		Force quadrature phase error interrupt.
		0	No effect.
		1	Force the interrupt.
1	PCE		Force position counter error interrupt.
		0	No effect.
		1	Force the interrupt.
0	Reserved	0	Always write as 0.

15.9.20 eQEP Status Register (QEPSTS)

Figure 15-40. eQEP Status (QEPSTS) Register (offset = 3Ah)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
UPEVNT	FIDF	QDF	QDLF	COEF	CDEF	FIMF	PCEF
R-1	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15-23. eQEP Status (QEPSTS) Register Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Always write as 0.
7	UPEVNT		Unit position event flag.
		0	No unit position event is detected.
		1	Unit position event is detected. Write 1 to clear.
6	FIDF		Direction on the first index marker. Status of the direction is latched on the first index event marker.
		0	Counter-clockwise rotation (or reverse movement) on the first index event.
		1	Clockwise rotation (or forward movement) on the first index event.
5	QDF		Quadrature direction flag.
		0	Counter-clockwise rotation (or reverse movement).
		1	Clockwise rotation (or forward movement).
4	QDLF		eQEP direction latch flag. Status of direction is latched on every index event marker.
		0	Counter-clockwise rotation (or reverse movement) on index event marker.
		1	Clockwise rotation (or forward movement) on index event marker.
3	COEF		Capture overflow error flag.
		0	Sticky bit, cleared by writing 1.
		1	Overflow occurred in eQEP capture timer (QCTMR).
2	CDEF		Capture direction error flag.
		0	Sticky bit, cleared by writing 1.
		1	Direction change occurred between the capture position event.
1	FIMF		First index marker flag.
		0	Sticky bit, cleared by writing 1.
		1	Set by first occurrence of index pulse.
0	PCEF		Position counter error flag. This bit is not sticky and it is updated for every index event.
		0	No error occurred during the last index transition.
		1	Position counter error.

15.9.21 eQEP Capture Timer Register (QCTMR)

Figure 15-41. eQEP Capture Timer (QCTMR) Register (offset = 38h)

15	0
QCTMR	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-24. eQEP Capture Timer (QCTMR) Register Field Descriptions

Bit	Field	Description
15-0	QCTMR	This register provides the time base for the edge capture unit.

15.9.22 eQEP Capture Period Register (QCPRD)

Figure 15-42. eQEP Capture Period (QCPRD) Register (offset = 3Eh)

15	0
QCPRD	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-25. eQEP Capture Period Register (QCPRD) Register Field Descriptions

Bit	Field	Description
15-0	QCPRD	This register holds the period count value between the last successive eQEP position events.

15.9.23 eQEP Capture Timer Latch Register (QCTMRLAT)

Figure 15-43. eQEP Capture Timer Latch (QCTMRLAT) Register (offset = 3Ch)

15	0
QCTMRLAT	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 15-26. eQEP Capture Timer Latch (QCTMRLAT) Register Field Descriptions

Bit	Field	Description
15-0	QCTMRLAT	The eQEP capture timer value can be latched into this register on two events: unit timeout event or reading the eQEP position counter.

15.9.24 eQEP Capture Period Latch Register (QCPRDLAT)

Figure 15-44. eQEP Capture Period Latch (QCPRDLAT) Register (offset = 42h)

15	0
QCPRDLAT	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 15-27. eQEP Capture Period Latch (QCPRDLAT) Register Field Descriptions

Bit	Field	Description
15-0	QCPRDLAT	The eQEP capture period value can be latched into this register on two events: unit timeout event or reading the eQEP position counter.

Analog To Digital Converter (ADC) Module

This chapter describes the analog to digital converter (ADC) interface module.

Topic **Page**

16.1 Overview

The main features of the ADC module are:

- Selectable 10-bit or 12-bit resolution
- Successive-approximation-register architecture
- Three conversion groups – Group1, Group2 and Event Group
- All three conversion groups can be configured to be hardware-triggered; group1 and group2 can also be triggered by software
- Conversion results are stored in a 64-word memory (SRAM)
  - These 64 words are divided between the three conversion groups and are configurable by software
  - Accesses to the conversion result RAM are protected by parity
- Selectable channel conversion order
  - Sequential conversions in ascending order of channel number, OR
  - User-defined channel conversion order with the Enhanced Channel Selection Mode
- Single or continuous conversion modes
- Embedded self-test logic for input channel failure detection (open / short to power / short to ground)
- Embedded calibration logic for offset error correction
- Enhanced Power-down mode
- External event pin (ADEVT) to trigger conversions
  - ADEVT is also programmable as general-purpose I/O
- Eight hardware events to trigger conversions

16.2 Introduction

This section presents a brief functional description of the analog-to-digital converter (ADC) module. Figure 16-1 illustrates the components of the ADC module.

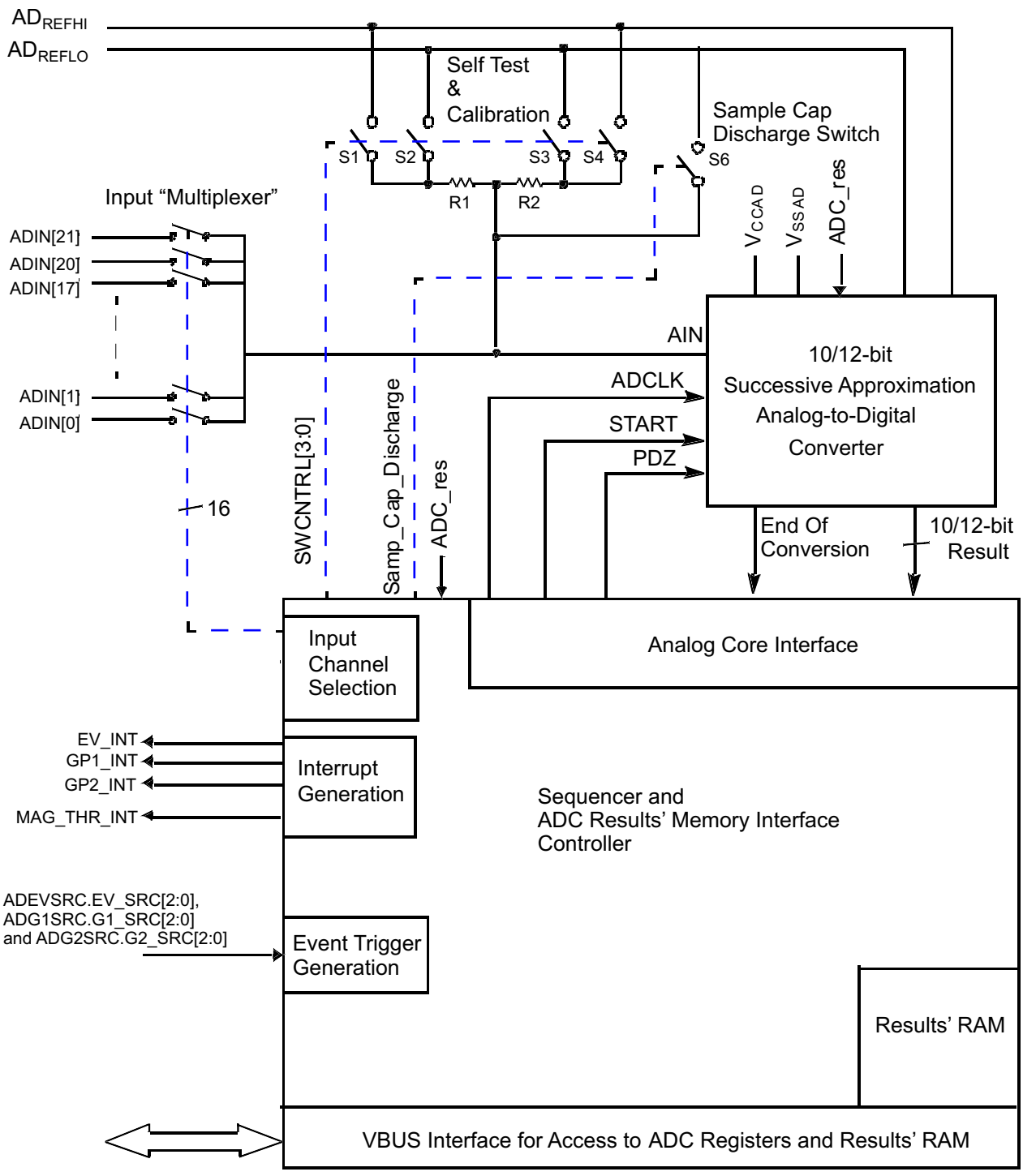
16.2.1 Input Multiplexor

The input multiplexor (MUX) connects the selected input channel to the AIN input of the ADC core. The ADC module supports up to 16 input channels. The sequencer selects the channel to be converted.

16.2.2 Self-Test and Calibration Cell

The ADC includes specific hardware that allows a software algorithm to detect open/short on an ADC analog input. It also allows the application program to calibrate the ADC. Also see Section 16.7.1 and Section 16.7.2.

Figure 16-1. ADC Block Diagram



16.2.3 Analog-to-Digital Converter Core

The ADC core is a combination voltage scaling, charge redistribution Successive Approximation Register (SAR) based analog-to-digital converter. The core can be configured for operation in 10-bit resolution (default) or 12-bit resolution. This is controlled by the sequencer logic. This selection applies to all conversions performed by the ADC module. It is not possible to convert some channels with a 12-bit resolution and some with a 10-bit resolution.

A single conversion from an analog input to a digital conversion result occurs in two distinct periods:

- Sampling Period:
  - The sequencer generates a START signal to the ADC core to signal the start of the sampling period.
  - The analog input signal is sampled directly on to the switched capacitor array during this period, providing an inherent sample-and-hold function.
  - The sampling period ends one full ADCLK after the falling edge of the START signal.
  - The sequencer can control the sampling period duration by configuring the conversion group's sample time control register (ADEVSAMP, ADG1SAMP, ADG2SAMP). This register controls the time for which the START signal stays high.
- Conversion Period:
  - The conversion period starts one full ADCLK after the falling edge of START.
  - One bit of the conversion result is output on each rising edge of ADCLK in the conversion period, starting with the most-significant bit first.
  - The conversion period is 12 ADCLK cycles in case of a 12-bit ADC, and is 10 ADCLK cycles in case of a 10-bit ADC.
  - The ADC core generates an End-Of-Conversion (EOC) signal to the sequencer at the end of the conversion period. At this time the complete 12-, or 10-bit conversion result is available.
  - The sequencer captures the ADC core conversion result output as soon as EOC is driven High.

The analog conversion range is determined by the reference voltages: AD_{REFHI} and AD_{REFLO}. AD_{REFHI} is the top reference voltage and is the maximum analog voltage that can be converted. An analog input voltage equal to AD_{REFHI} or higher results in an output code of 0x3FF for 10-bit resolution and 0xFFF for 12-bit resolution. AD_{REFLO} is the bottom reference voltage and is the minimum analog voltage that can be converted. Applying an input voltage equal to AD_{REFLO} or lower results in an output code of 0x000. Both AD_{REFHI} and AD_{REFLO} must be chosen not to exceed the analog power supplies: V_{CCAD} and V_{SSAD}, respectively. Input voltages between AD_{REFHI} and AD_{REFLO} produce a conversion result given by Equation 30 for 10-bit resolution and by Equation 31 for 12-bit resolution.

$$DigitalResult = \frac{1024 \; x \; (InputVoltage - AD_{REFLO})}{AD_{REFHI} - AD_{REFLO}} - 0.5 \tag{30}$$

$$DigitalResult = \frac{4096 \, x (InputVoltage - AD_{REFLO})}{(AD_{REFHI} - AD_{REFLO})} - 0.5 \tag{31}$$

16.2.4 Sequencer

The sequencer coordinates the operations of the ADC, including the input multiplexor, the ADC core, and the result memory. In addition, the logic of the sequencer sets the status register flags when the conversion is ongoing, stopped, or finished.

All the features of the sequencer are discussed in detail in the following sections.

16.2.5 Conversion Groups

Several applications require groups of channels to be converted using a single trigger source for example. There could also be some groups of channels identified which require a specific setting of the acquisition time. The ADC module supports three conversion groups for this purpose – Group1, Group2 and the Event Group.

Any of the available analog input channels can be assigned to any of the conversion groups. This also allows a particular channel to be repeatedly sampled by selecting it in multiple groups. There is an inherent priority scheme used when multiple conversion groups are triggered at once. The Event Group is the highest-priority, followed by the Group1 and then the Group2.

The Event Group is always hardware event-triggered. Group1 and Group2 are software-triggered by default and can be configured to be hardware-, or event-triggered as well. The triggering of conversions in each group is discussed in Section 16.3.6.

Each conversion group has a separate set of control registers to:

- Select the input channels to be converted
- Configure the mode of conversion: single conversion sequence or continuous conversions
- Configure the input channel sampling time
- Configure the interrupt request generation conditions

16.3 Basic Features and Usage of the ADC

This section describes the usage of the basic features of the ADC module.

16.3.1 How to Select Between 12-bit and 10-bit Resolutions

The 10_12_BIT field of the ADC Operating Mode Control Register (ADOPMODECR) configures the ADC to be in 10-bit or 12-bit resolution mode.

- If 10_12_BIT = 0, the module is in 10-bit resolution mode. This is the default mode of operation.
- If 10_12_BIT = 1, the module is in 12-bit resolution mode.

16.3.2 How to Set Up the ADCLK Speed

The ADC sequencer generates the clock for the ADC core, ADCLK. The ADC core uses the ADCLK signal for its timing. The ADCLK is generated by dividing down the input clock to the ADC module, which is the VBUSP interface clock, VCLK. A 5-bit field (PS) in the ADC Clock Control Register (ADCLOCKCR) is used to divide down the VCLK by 1 up to 32. The ADCLK valid frequency range is specified in the device datasheet.

$f_{ADCLK} = f_{VCLK} / (PS + 1)$

The maximum frequency for ADCLK is specified in the device datasheet.

16.3.3 How to Set Up the Input Channel Acquisition Time

The signal acquisition time for each group is separately configurable using the ADG1SAMP[11:0], ADG2SAMP[11:0], and ADEVSAMP[11:0] registers.

The acquisition time is specified in terms of ADCLK cycles and ranges from a minimum of 2 ADCLK cycles to a maximum of 4098 ADCLK cycles.

For example, Group1 acquisition time, t_{ACQG1} = G1SAMP[11:0] + 2, in ADCLK cycles.

The minimum acquisition time is specified in the device datasheet. This time also depends on the impedance of the circuit connected to the analog input channel being converted. See the *ADC Source Impedance for Hercules™ ARM® Safety MCUs Application Report* (SPNA118).

16.3.4 How to Select an Input Channel for Conversion

The ADC module needs to be enabled first before selecting an input channel for conversion. The ADC module can be enabled by setting the ADC EN bit in the ADC Operating Mode Control Register (ADOPMODECR). Multiple input channels can be selected for conversion in each group. Only one input channel is converted at a time. The channels to be converted are configured in one or more of the three conversion groups' channel selection registers. Channels to be converted in Group1 are configured in the Group1 Channel-Select Register (ADG1SEL), those to be converted in Group2 are configured in the Group2 Channel-Select Register (ADG2SEL), and those to be converted in the Event Group are configured in the Event Group Channel-Select Register (ADEVSEL).

The description in this section only refers to the case when the enhanced channel selection mode is not enabled. Input channel selection in the enhanced channel selection mode is defined in Section 16.4.

16.3.5 How to Select Between Single Conversion Sequence or Continuous Conversions

Each group has its own mode control register. The MODE field of these control registers allow the application to select between a single conversion sequence or continuous conversion mode.

NOTE: Selecting continuous conversion mode for all three groups

All three conversion groups cannot be configured to be in a continuous conversion mode. If the application configures the group mode control registers to enable continuous conversion mode for all three groups, then the Group2 will be automatically be configured to be in a single conversion sequence mode.

With conversions ongoing in continuous conversion mode, if the MODE field of a group is cleared, then that group switches to the single conversion sequence mode. Conversions for this group will stop once all channels selected for that group have been converted.

16.3.6 How to Start a Conversion

The conversion groups Group1 and Group2 are software-triggered by default. A conversion in these groups can be started just by writing the desired channels to the respective Channel-Select Registers. For example, in order to convert channels 0, 1, 2, and 3 in Group1 and channels 8, 9, 10, and 11 in Group2, the application just has to write 0x0000000F to ADG1SEL and 0x00000F00 to ADG2SEL. The ADC module will start by servicing the group that was triggered first, Group1 in this example.

The conversions for all groups are performed in ascending order of the channel number. For the Group1 the conversions will be performed in the order: channel 0 first, followed by channel 1, then channel 2, and then channel 3. The Group2 conversions will be performed in the order: channels 8, 9, 10, and 11.

The Event Group is only hardware-triggered. There are up to eight hardware event trigger sources defined for the ADC module. Check the device datasheet for a complete listing of these eight hardware trigger options.

The trigger source to be used needs to be configured in the ADEVSRC register. Similar registers also exist for the Group1 and Group2 as these can also be configured to be event-triggered.

The polarity of the event trigger is also configurable, with a falling edge being the default.

An Event Group conversion starts when at least one channel is selected for conversion in this group, and when the defined event trigger occurs.

If any conversion group is configured to be in a continuous conversion mode, then it needs to only be triggered once. All the channels selected for conversion in that group will be converted repeatedly.

16.3.7 How to Know When the Group Conversion is Completed

Each conversion group has a status flag to indicate when its conversion has ended. See ADEVSR, ADG1SR, and ADG2SR. This bit is set when a conversion sequence for a group ends. This bit does is always set if a group is configured for continuous conversions.

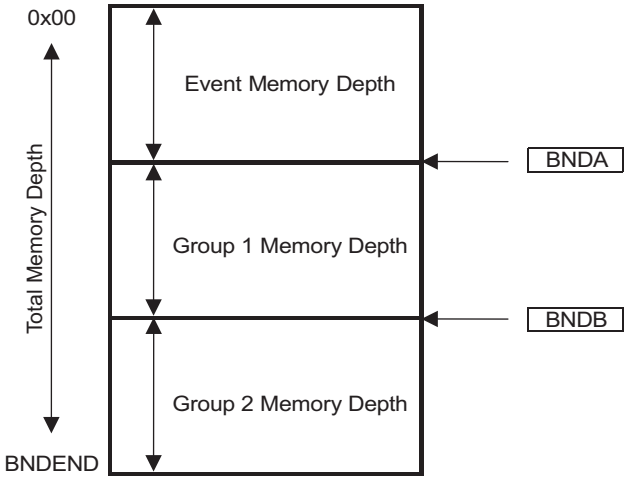
16.3.8 How Results are Stored in the Results' Memory

The ADC stores the conversion results in three separate memory regions in the ADC Results' RAM, one region for each group. Each memory region is a stack of buffers, with each buffer capable of holding one conversion result. The number of buffers allocated for each group is programmed by configuring the ADC module registers ADBNDCR and ADBNDEND.

ADBNDCR contains two 9-bit pointers BNDA and BNDB. BNDA, BNDB, and BNDEND are used to partition the total memory available into three memory regions as shown in Figure 16-2. Both BNDA and BNDB are pointers referenced from the start of the results' memory. BNDA specifies the number of buffers allocated for the Event Group conversion results in units of two buffers; BNDB specifies the number of buffers allocated for the Event Group plus Group1 in units of two buffers. Refer to Section 16.10.20 for more details on configuring the ADC results' memory.

ADBNDEND contains a 3-bit field called BNDEND that configures the total memory available. The ADC module can support up to 1024 buffers. The device supports a maximum of 64 buffers for both the ADC modules.

Figure 16-2. FIFO Implementation



- Number of buffers for Event Group = 2 × BNDA
- Number of buffers for Group1 = 2 × (BNDB – BNDA)
- Number of buffers for Group2 = Total number of buffers – 2 × BNDB

16.3.9 How to Read the Results from the Results' Memory

The CPU can read the conversion results in one of two ways:

1. By using the conversion results memory as a FIFO queue
2. By accessing the conversion results memory directly

16.3.9.1 Reading Conversion Results from a FIFO

The conversion results for each group can be accessed via a range of addresses provided to facilitate the use of the ARM Cortex-R4 CPU's Load-Multiple (LDM) instruction. A single read performed using the LDR instruction can also be used to read out a single conversion result. The results are read out from the group's memory region as a FIFO queue by reading from any location inside this address range. The conversion result that got stored first gets read first. A result that is read from the memory in this method is removed from the memory. For example, a read from any address in the range ADEVBUFFER (offset 90h to AFh) pulls out one conversion result from the Event Group memory.

Figure 16-3. Format of Conversion Result Read from FIFO, 12-bit ADC

Offset Address Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x90 to 0xAF ADEVBUFFER	EV_ EMPTY	Reserved										EV_CHID				
	Reserved				EV_DR											
0xB0 to 0xCF ADG1BUFFER	G1_ EMPTY	Reserved										G1_CHID				
	Reserved				G1_DR											
0xD0 to 0xEF ADG2BUFFER	G2_ EMPTY	Reserved										G2_CHID				
	Reserved				G2_DR											

Figure 16-4. Format of Conversion Result Read from FIFO, 10-bit ADC

Offset Address Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x90 to 0xAF ADEVBUFFER	Reserved															
	EV_ EMPTY	EV_CHID					EV_DR									
0xB0 to 0xCF ADG1BUFFER	Reserved															
	G1_ EMPTY	G1_CHID					G1_DR									
0xD0 to 0xEF ADG2BUFFER	Reserved															
	G2_ EMPTY	G2_CHID					G2_DR									

Option to read channel id along with conversion result:

The application has an option to read the channel id along with the conversion result. This is controlled by the CHID field of the group's mode control register. If the option to read the channel id is not selected, the channel id field of the conversion result reads as zeros.

Protection against reading from empty FIFO:

There is also a hardware mechanism to protect the application from reading past the number of new conversion results held in the FIFO. Once all available conversion results have been read out of the FIFO by the application, a subsequent read from the FIFO causes the mechanism to indicate that the FIFO is empty by setting the EMPTY field.

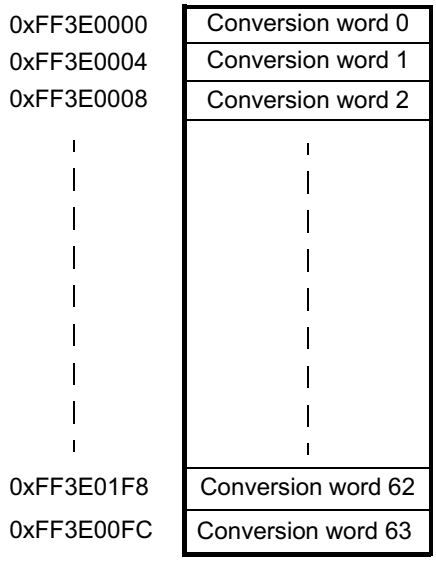
Debug / Emulation Support:

For debug purposes, each conversion group also provides an address that the application can read from for extracting the group's conversion results. However, no status flags for a conversion group are affected by reading from these emulation buffer addresses. For example, reading from ADEVEMUBUFFER (offset 0xF0) returns the next result in the Event Group buffer but does not actually remove that result from the buffer or change the amount of data held in the buffer.

16.3.9.2 Reading Conversion Results Directly from the Conversion Results' Memory

The conversion result memory is part of the device's memory map. The base address for the ADC result memory is 0xFF3E0000.

Figure 16-5. ADC Memory-Mapping



The application can identify the address ranges for each of the three memory regions for the three conversion groups after performing the segmentation as described in Section 16.3.8. It is up to the application to read the desired results from the three conversion groups. The formats of the conversion results when reading from RAM directly are shown in Figure 16-6 and Figure 16-7.

Figure 16-6. Format of Conversion Result Directly Read from ADC RAM, 12-bit ADC

	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
ADC RAM address	Reserved															channel id [4]
	channel id [3-0]						12-bit conversion result									

Figure 16-7. Format of Conversion Result Directly Read from ADC RAM, 10-bit ADC

	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
ADC RAM address	Reserved															
	Rsvd	channel id [4-0]					10-bit conversion result									

Note that there is no EMPTY field to protect the application from reading data that has been previously read.

Each group does have a separate register which holds the address in the group's result memory where the ADC will write the next conversion result. These are the ADEVRAMWRADDR, ADG1RAMWRADDR, and ADG2RAMWRADDR registers. The application can use this information to calculate how many valid conversion results are available to be read.

Benefit of reading conversion results directly from ADC RAM:

The application does not have to read out conversion results sequentially as in the case of reading from a FIFO. As a result, the application can selectively read the conversion results for any particular input channel of interest without having to read other channels' conversion results.

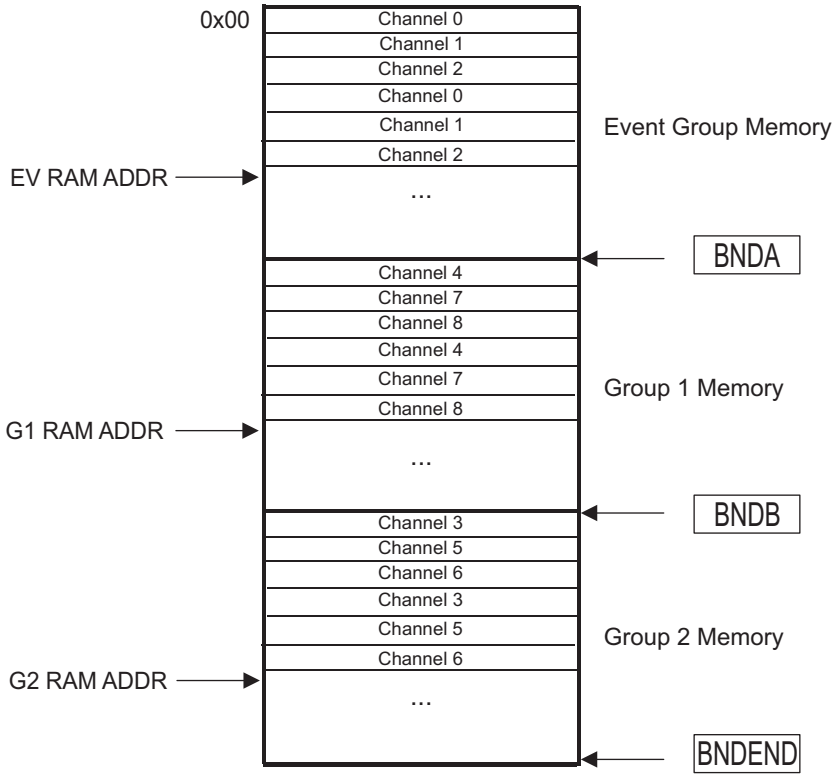
16.3.9.3 Example

Suppose that channels 0, 1, and 2 are selected for conversion in the Event Group, channels 4, 7, and 8 are selected for conversion in group 1, and channels 3, 5, and 6 are selected for conversion in group 2. The conversion results will get stored in the three memory regions as shown in Figure 16-8.

Suppose that the CPU wants to read out the results for the Event Group from a FIFO queue. The CPU needs to read from any address in the range ADEVBUFFER (offset 0x90 to 0xAF) multiple times, or do a "load multiple" from this range of addresses. This will cause the ADC to return the results for channel 0, then channel 1, then channel 2, then channel 0, and so on for each read access to this address range.

Now suppose that the application wants to read out the results for the group 1 from the RAM directly. The conversion results for the group 1 are accessible starting from address ADC RAM Base Address + BNDA. Also, it is known that the first result at this address is for the input channel 4, the next one is for input channel 7, and so on. So the application can selectively read the conversion results for only one channel if so desired.

Figure 16-8. Conversion Results Storage



16.3.10 How to Stop a Conversion

A group's conversion can be stopped by clearing the group's channel select register.

16.3.11 Example Sequence for Basic Configuration of ADC Module

The following sequence is necessary to configure the ADC to convert channels 0, 2, 4, and 8 in single-conversion mode using Group1:

1. Write 0 to the Reset Control Register (ADRSTCR) to release the module from the reset state.
2. Write 1 to the ADC_EN bit of the Operating Mode Control Register (ADOPMODECR) to enable the ADC state machine.
3. Configure the ADCLK frequency by programming the desired divider into the Clock Control Register (ADCLOCKCR).
4. Configure the acquisition time for the group that is to be used. For example, configure the Group1 Sampling Time Control Register (ADG1SAMP) to set the acquisition time for Group1.
5. Select the channels that need to be converted in Group1 by writing to the Group1 Channel Select Register (ADG1SEL). In this example, a value of 0x115 needs to be written to ADG1SEL in order to select channels 0, 2, 4, and 8 for conversion in Group1.
   - The ADC sequencer will start the Group1 conversions as soon as the write to the ADG1SEL register is completed.
6. Wait for the G1_END bit to be set in the Group1 Conversion Status Register (ADG1SR). This bit gets set when all the channels selected for conversion in Group1 are converted and the results are stored in the Group1 memory.
7. Read the conversion results by reading from the Group1 FIFO access location (ADG1BUFFER) or by reading directly from the Group1 results' memory.

16.4 Advanced Conversion Group Configuration Options

Figure 16-9 shows the operating mode control registers and the status registers for each of the three conversion groups. The register addresses shown are offsets from the base address. The ADC register frame base address is FFF7 C000h.

Figure 16-9. ADC Groups' Operating Mode Control and Status Registers

Offset Address Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x010 ADEVMODECR	Reserved															No Reset On ChnSel
	Reserved						EV_ DATA_FMT		Reserved		EV_ CHID	OVR_ EV_ RAM_ IGN	Rsvd	EV_ 8BIT	EV_ MODE	FRZ_ EV
0x014 ADG1MODECR	Reserved															No Reset On ChnSel
	Reserved						G1_ DATA_FMT		Reserved		G1_ CHID	OVR_ G1_ RAM_ IGN	Rsvd	G1_ 8BIT	G1_ MODE	FRZ_ G1
0x018 ADG2MODECR	Reserved															No Reset On ChnSel
	Reserved						G2_ DATA_FMT		Reserved		G2_ CHID	OVR_ G2_ RAM_ IGN	Rsvd	G2_ 8BIT	G2_ MODE	FRZ_ G2
0x06C ADEVSR	Reserved															
	Reserved												EV_ MEM_ EMPTY	EV_ BUSY	EV_ STOP	EV_ END
0x070 ADG1SR	Reserved															
	Reserved												G1_ MEM_ EMPTY	G1_ BUSY	G1_ STOP	G1_ END
0x074 ADG2SR	Reserved															
	Reserved												G2_ MEM_ EMPTY	G2_ BUSY	G2_ STOP	G2_ END
0x19C ADEVCURRCOUNT	Reserved															
	Reserved											EV_CURRENT_COUNT				
0x1A0 ADEVMAXCOUNT	Reserved															
	Reserved											EV_MAX_COUNT				
0x1A4 ADG1CURRCOUNT	Reserved															
	Reserved											G1_CURRENT_COUNT				
0x1A8 ADG1MAXCOUNT	Reserved															
	Reserved											G1_MAX_COUNT				
0x1AC ADG2CURRCOUNT	Reserved															
	Reserved											G2_CURRENT_COUNT				
0x1B0 ADG2MAXCOUNT	Reserved															
	Reserved											G2_MAX_COUNT				

The following sections describe each of these group configuration options separately.

16.4.1 Group Trigger Options

The Group1 and Group2 operating mode control registers have an extra control bit: HW_TRIG. This bit configures the group to be hardware event-triggered instead of software-triggered, which is the default.

When a group is configured to be event-triggered, the group conversion starts when at least one channel is selected for conversion in this group, and when the defined event trigger occurs. The event trigger source is defined for each group in the ADEVSRC, ADG1SRC, and the ADG2SRC registers. The actual connections used as the event trigger sources are defined in the device datasheet for both the ADC modules.

16.4.2 Analog Input Channel Selection Mode Options

The ADC modules on this device support two different modes for selecting the analog input channel to be converted. These are the sequential channel selection mode (default), and the enhanced channel selection mode. These two modes are now described:

16.4.2.1 Sequential Channel Selection Mode

This is the default mode and allows the ADC module to be used in a backwards compatible mode to the ADC module on other Hercules™ ARM® Safety MCUs from Texas Instruments. As discussed in Section 16.3.4, an analog input channel can be selected for conversion in one or more conversion groups by setting the bit corresponding to that channel number in the group's channel select register.

16.4.2.2 Enhanced Channel Selection Mode

There are some important concepts related to the enhanced channel selection mode. These are defined first:

- Look-Up Table

  This is a 32-word deep memory-mapped region used to define the analog input channel number to be converted. The look-up tables (LUTs) for the three groups are stacked together so that the entire LUT occupies 96 words. Each word is aligned on a 32-bit boundary. The LUTs for ADC start at 0xFF3E2000.

- Conversion Group Sub-Sequence

  A group sub-sequence is defined as the conversion for a set of channels that is converted on each conversion trigger. The number of channels selected for conversion in a group sub-sequence is defined by the number of bits that are set in the group's channel select register. For example, setting bits 0, 1, 29, and 31 in ADG1SEL means that each Group1 conversion sub-sequence consists of 4 conversions.

- LUT Index

  A "CURRENT_COUNT" register for each group is maintained as an index into that group's LUT. This register increments each time a channel conversion is completed. Therefore, as its name suggests, a read from this register returns the number of conversions completed since the last write to the group's channel select register. The CURRENT_COUNT register resets to all zeros under any of the following conditions:

  1. The ADC peripheral is reset via a global peripheral reset
  2. The ADC peripheral is reset via the ADC Reset Control Register, see Section 16.10.1
  3. The CURRENT_COUNT becomes equal to the MAX_COUNT defined for that conversion group
  4. The application writes zeros to the CURRENT_COUNT register
  5. The conversion group's result RAM is reset

- Maximum Number of Conversions

  A "MAX_COUNT" register for each conversion group stores the maximum number of conversions to be performed before the index into a group's LUT is reset to zero. This register can be programmed to a value between 0 and 31. It is recommended to program the MAX_COUNT register with a value that is one less than a multiple of the number of channels in that group's conversion sub-sequence (number of bits that are set in the group's channel select register).

16.4.2.2.1 Look-Up Table Details

As described earlier, each conversion group has a look-up table (LUT) that is used when the enhanced channel selection mode is enabled. This look-up table starts at an offset of 8kB from the base of the ADC results RAM. The LUT holds 32 entries for each of the three conversion groups. The first 32 entries are for the event group, the next 32 entries are for Group1 and the last 32 entries are for Group2. Figure 16-10 shows an example LUT entry for the Event group.

Figure 16-10. Example Look-Up Table Entry

31			16
Reserved			
R-0			

15	13	12	8	7	5	4	0
Reserved		EV_EXT_CHN_MUX_SEL		Reserved		EV_INT_CHN_MUX_SEL	
R-0		R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-1. ADC Look-Up Table Field Descriptions

Bit	Field	Value	Description
31–13	Reserved	0	Reads return 0. Writes have no effect.
12–8	EV_EXT_CHN_MUX_SEL		Do not write to this bit field.
7–5	Reserved	0	Reads return 0. Writes have no effect.
4–0	EV_INT_CHN_MUX_SEL		This field defines the internal analog mux select that is output from the ADC module when the Event group CURRENT_COUNT register points to this LUT entry, and when the Event group conversion is triggered with the enhanced channel selection mode enabled. This can be a value between 0 and 31, which corresponds to the internal analog input channel number between 0 and 31. Note that this device only supports 16 input channels. If the application configures an unavailable channel number in the EV_INT_CHN_MUX_SEL field, the ADC will still perform the conversion and the result will be indeterminate.

16.4.3 Single or Continuous Conversion Modes

The EV_MODE, G1_MODE, and G2_MODE bits are used to select between either single or continuous conversion mode for each of the three groups.

16.4.3.1 Single Conversion Mode

A conversion group configured to be in single-conversion mode gets serviced only once by the ADC for each group trigger. The trigger can be a software trigger as in the case of Group1 and Group2 by default, or it could be a hardware event trigger as in the case of the Event Group or Group1 or Group2.

The entire conversion sequence, from the acceptance of the group conversion request to the end of the last channel's conversion, is flagged for each group by the corresponding BUSY bit in that group's status register. After single-conversion mode is started, the BUSY bit is read as 1 until the conversion of the last channel is complete. The END bit for the group is set once all the channels in that group are converted.

For example, say channels 0, 2, 4, and 6 are selected for conversion in Group1 in single-conversion mode. When the Group1 gets serviced, the ADC will start conversion for channel 0, then channel 2, then channel 4, and then channel 6. It will then stop servicing the Group1, set the G1_END status bit, and look to service the Event Group or the Group2, if required.

16.4.3.2 Continuous Conversion Mode

A conversion group configured to be in continuous-conversion mode gets serviced by the ADC continuously. The group still needs to be triggered appropriately for the first conversion to start. The conversions are performed continuously thereafter.

The entire conversion sequence, from the acceptance of the group conversion request to the end of the last channel's conversion, is flagged for each group by the corresponding BUSY bit in that group's status register. After continuous-conversion mode is started, the BUSY bit is read as 1 as long as the continuous-conversion mode for this group is selected.

As an example, say the channels 0, 2, 4, and 6 are selected for conversion in Group1, now in continuous-conversion mode. When the Group1 gets serviced, the ADC will complete conversions for channels 0, 2, 4 and 6, and then look to service the Event Group or the Group2. Once it is done servicing the Event Group or the Group2, it will return to service the Group1 again. The Group1 does not need to be triggered again for the repeated conversion.

NOTE: Configuring all conversion groups in continuous conversion mode

All the three groups cannot operate in continuous-conversion mode at the same time. If the application program configures all three groups to be in continuous-conversion mode, the Group2 is automatically reset to single-conversion mode, and the G2_MODE bit in the ADG2MODECR register is cleared to reflect the single-conversion mode of Group2.

16.4.4 Conversion Group Freeze Capability

The ADC module has an inherent priority order between the three conversion groups. This group priority determines the order of conversion in case multiple groups are triggered. The priority of conversions between the three groups in descending order is:

1. Event Group
2. Group1
3. Group2

Examples of conversion group priority:

- If an Event Group conversion is ongoing in single conversion sequence mode and Group2 and Group1 conversions are requested, then the ADC will finish conversion of channels selected in Event Group, then switch over to converting channels selected in Group1, and then convert channels selected in Group2.
- If Group1 conversions are ongoing in continuous conversion mode and Group2 conversion is requested, then the ADC will complete converting the current channel for Group1 and switch over to converting channels selected in Group2. The new conversion request for Group2 has a higher priority than the pending continuous conversion request for Group1.

The conversion group freeze capability allows the application to override this default priority between the conversion groups. Enabling the freeze capability allows the ADC to freeze a higher-priority conversion group's conversions whenever there is a request for conversion in another (lower-priority) group.

For example, setting the FRZ_EV bit in the ADEVMODECR register will allow the ADC to freeze ongoing Event Group conversions whenever there is a pending request, or a new request for a Group1 or Group2 conversion. The conversions for the Event Group will be frozen as long as the Group1 or Group2 conversions are active. Once the Group1 or Group2 conversions are completed, the Event Group conversions start from where they were frozen.

While a group's conversions are frozen, the group's STOP status bit is set. This bit is cleared once the group's conversions are restarted.

16.4.5 Conversion Group Memory Overrun Option

An overrun condition occurs when the ADC module tries to store more conversion results to a group's results' memory which is already full. In this case, the ADC allows two options.

If the OVR_RAM_IGN bit in the group's operating mode control register (ADEVMODECR, ADG1MODECR, ADG2MODECR) is set, then the ADC module ignores the contents of the group's results' memory and wraps around to overwrite the memory with the results of new conversions.

If the OVR_RAM_IGN bit is not set, then the application program has to read out the group's results' memory upon an overrun condition; only then can the ADC continue to write new results to the memory.

16.4.6 Response on Writing Non-Zero Value to Conversion Group's Channel Select Register

If the application writes a non-zero value to a group's channel select register while that group's conversions are already being serviced, then that group's conversions will be restarted with the new configuration programmed in the channel select registers.

The following rules apply in terms of the effect on the ADC conversion sequence:

- If the new conversion request comes from the same group as the ongoing conversion, then the ongoing conversion will be stopped in whichever stage it is in, and the new sequence of conversions will be started.
- If the new conversion request comes from a separate group, then the ongoing channel's conversion will be completed before starting the new sequence of conversions.

The following rules apply in terms of the effect on the group's results memory:

- If a group conversion is ongoing or is frozen, writing a non-zero value to the group's channel select register will also reset its results FIFO. This does not clear the contents of the results FIFO; only the ADC module is allowed to overwrite the FIFO's contents with new conversion results starting from the first location.
- If the group conversion is completed (<GRP>_END flag is set), or the group is not being used, then writing a non-zero value to the group's channel select register will either be reset or not depending on the value of the NoResetOnChnSel bit for that group (ADEVMODECR, ADG1MODECR, ADG2MODECR).
  - If the NoResetOnChnSel bit is 0, then the group's FIFO will be reset.
  - If the NoResetOnChnSel bit is 1, then the group's FIFO will not be reset.

16.4.7 Conversion Result Size on Reading: 8-bit, 10-bit, or 12-bit

Some applications do not need the full 12-bit resolution of the ADC modules on the device and can work with 8-bit or 10-bit conversion results.

16.4.7.1 ADC Configured in 12-bit Resolution

The mode control register for each conversion group contains a field called DATA_FMT, which defines the format of the conversion result read out of the result RAM, when accessed as a FIFO.

The DATA_FMT field is encoded as follows:

- If DATA_FMT = 00, the complete 12-bit conversion result is read out of the FIFO.
- If DATA_FMT = 01, the 12-bit conversion result is right-shifted by 2 and the resulting 10-bit result is read out of the FIFO.
- If DATA_FMT = 10, the 12-bit conversion result is right-shifted by 4 and the resulting 8-bit result is read out of the FIFO.

This control field is not effective when the application chooses to access the conversion result memory directly. In that case, the application can choose to mask off the number of bits as required.

16.4.7.2 ADC Configured in 10-bit Resolution

The DATA_FMT field is not effective in this mode and the application has the choice to read either the full 10-bit conversion result or an 8-bit conversion result. This is controlled by the 8BIT field of the group's operating mode control register.

- If 8BIT = 0, the complete 10-bit conversion result is read out of the FIFO.
- If 8BIT = 1, the 10-bit conversion result is right-shifted by 2 and the resulting 8-bit result is read out of the FIFO.

16.4.8 Option to Read Group Channel ID Along With Conversion Result

The ADC module allows the application program to also read out the analog input channel number along with its conversion result. This capability is enabled by setting the CHID bit in the group's operating mode control register.

- If CHID = 0, the bits [14-10] are forced to 00000 when the conversion results are read out from the group's results' FIFO.
- If CHID = 1, the bits [14-10] in the group's results' memory contain the input channel number to which the conversion result belongs.

NOTE: Actual Storage of Channel ID

Regardless of whether the CHID bit is set or not, the channel number is **always stored** in the memory along with the conversion result. The CHID bit only affects whether the channel number is available with the conversion result **when the group's memory is read**. Therefore, the CHID bit for a group can be changed dynamically without affecting that group's ongoing conversions.

16.5 ADC Module Basic Interrupts

This section describes the basic interrupts generated by the ADC module.

16.5.1 Group Conversion End Interrupt

The ADC module sets the group's conversion end flag (EV_END, G1_END, or G2_END) in that group's interrupt flag register (ADEVINTFLG, ADG1INTFLG, ADG2INTFLG) when all the channels selected for conversion in that group are converted. This causes a group conversion end interrupt to be generated, if this interrupt is enabled by setting the group's END_INT_EN control bit (EV_END_INT_EN, G1_END_INT_EN, or G2_END_INT_EN).

This interrupt can be easily used for conversion groups configured to be in the single-conversion mode. The application program can read out the conversion results, change the group's configuration if necessary, and restart the conversions by triggering the group from within the interrupt service routine.

For groups configured to be in continuous conversion mode, this interrupt condition is not practical as the conversions are always in progress. In this case, the Group Memory Threshold Interrupt is more practical as the application can allow a programmable number of conversion results to accumulate before interrupting the CPU.

16.5.2 Group Memory Threshold Interrupt

The ADC module has the ability to generate an interrupt for a fixed number of conversions for each group. A group memory threshold register determines how many conversion results must be in a group's memory region before the CPU is interrupted. This feature can be used to significantly reduce the CPU load when using interrupts for reading the conversion results.

The group's threshold register needs to be configured before the group conversions are triggered. This threshold register value behaves like a down-counter, which decrements each time the ADC writes a conversion result to this group's memory. This counter is incremented each time the application program reads a conversion result from the results' memory by accessing the FIFO queue. Simultaneous read (by application program) and write (by ADC module) operations from the group's results' memory leave the threshold counter unchanged.

The threshold counter can decrement past 0 and become negative. It always increments back to its original value when the memory region is emptied. To determine how many samples are in the memory region at a given moment, the threshold counter can be subtracted from the originally configured threshold count.

Whenever the threshold counter transitions from +1 to 0, it sets the group's threshold interrupt flag, and the CPU is interrupted if the group's threshold interrupt is enabled. The CPU is expected to clear the interrupt flag after reading the conversion results from the memory.

The interrupt flag is not set when the threshold counter stays at 0 or transitions from -1 to 0.

16.5.3 Group Memory Overrun Interrupt

An interrupt can be generated for each group if the number of ADC conversions for that group exceed the number of buffers allocated for that conversion group. Alternatively, the application program can set the group's OVR_RAM_IGN bit and allow the ADC module to overwrite the group's results' memory contents with new conversion results.

16.6 ADC Magnitude Threshold Interrupts

The ADC allows up to three magnitude threshold interrupts to be generated. The comparison parameters are programmed via the ADC Magnitude Compare Interrupt x Control Register (ADMAGINTxCR).

16.6.1 Magnitude Threshold Interrupt Configuration

The following fields are configurable for each of the three available magnitude threshold interrupts:

1. CHN_THR_COMP: Specifies whether to compare two channels' conversion results, or to compare a channel's conversion result to a programmable threshold value. A value of 0 will select the programmable threshold to be compared, and a value of 1 will select the conversion result of the channel identified by the COMP_CHID field to be compared.
2. MAG_CHID: Specifies the channel number from 0 to 31 whose conversion result needs to be monitored.
3. COMP_CHID: Specifies the channel number from 0 to 31 whose last conversion result is used for the comparison with the conversion result of the channel being monitored.
4. MAG_THR: Specifies the value for comparison with the conversion result of the channel identified by the MAG_CHID field.
5. CMP_GE_LT: Specifies whether the conversion result of the channel identified by MAG_CHID is compared to be "greater than or equal to", or "less than" the reference value. The reference value can be the conversion result of another channel identified by the COMP_CHID field, or it could be a threshold value specified in the MAG_THR field. A value of 0 in the CMP_GE_LT field indicates a "less than" comparison and a value of 1 indicates a "greater than or equal to" comparison.

16.6.2 Magnitude Threshold Interrupt Comparison Mask Configuration

There is also a separate comparison mask register (ADMAGxMASK) for each of the three magnitude threshold interrupts. This register is used to specify the bits that are masked off for the sake of the comparison. For example, the lower 4 bits of the conversion result can be masked off by writing 0xF to the interrupt comparison mask register, allowing a gross comparison to be made. By default, the full 10/12-bit conversion results are compared.

16.6.3 Magnitude Threshold Interrupt Enable / Disable Control

Each of the three magnitude interrupts also have separate interrupt enable set (ADMAGINTENASET) and clear (ADMAGINTENACLR) registers. These are used to respectively enable and disable that particular magnitude threshold interrupt from being generated. To enable a magnitude threshold interrupt, write a 1 to the corresponding bit of the interrupt enable set register. Conversely, to disable a magnitude threshold interrupt, write a 1 to the corresponding bit of the interrupt enable clear register.

16.6.4 Magnitude Threshold Interrupt Flags

There is a separate magnitude interrupt flag register (ADMAGINTFLG) that holds the flags for these three interrupts. This flag gets set whenever the comparison condition for the corresponding interrupt is met. A magnitude threshold interrupt is generated if the corresponding flag is set inside the flag register, and the interrupt generation is enabled. This flag can be cleared by writing a 1 to the flag or by reading from the interrupt offset register in case of this interrupt being the current highest-priority pending interrupt.

16.6.5 Magnitude Threshold Interrupt Offset Register

It is possible to have multiple magnitude threshold interrupts pending at the same time. The magnitude threshold interrupt offset register (ADMAGINTOFF) holds the index of the currently pending highest priority magnitude threshold interrupt. The magnitude threshold interrupt 1 has the highest priority while the magnitude threshold interrupt 3 has the lowest priority. This is a read-only register and returns zeros if none of the magnitude threshold interrupts are pending. Writes to this register have no effect.

A read from this register updates the register to the next highest-priority pending magnitude threshold interrupt. This read also clears the corresponding flag from the magnitude threshold interrupt flag register. However, a read from the magnitude threshold interrupt offset register in emulation mode does not affect the interrupt flag register or the interrupt offset register.

16.7 ADC Special Modes

The ADC module supports some special modes for diagnostics and power saving purposes.

16.7.1 ADC Error Calibration Mode

The application program can activate a calibration sequence any time self-test mode is disabled (SELF_TEST = 0). This calibration sequence includes the conversion of an embedded calibration reference voltage followed by the calculation of an offset error correction value.

NOTE: Disable Self-Test Mode Before Calibration

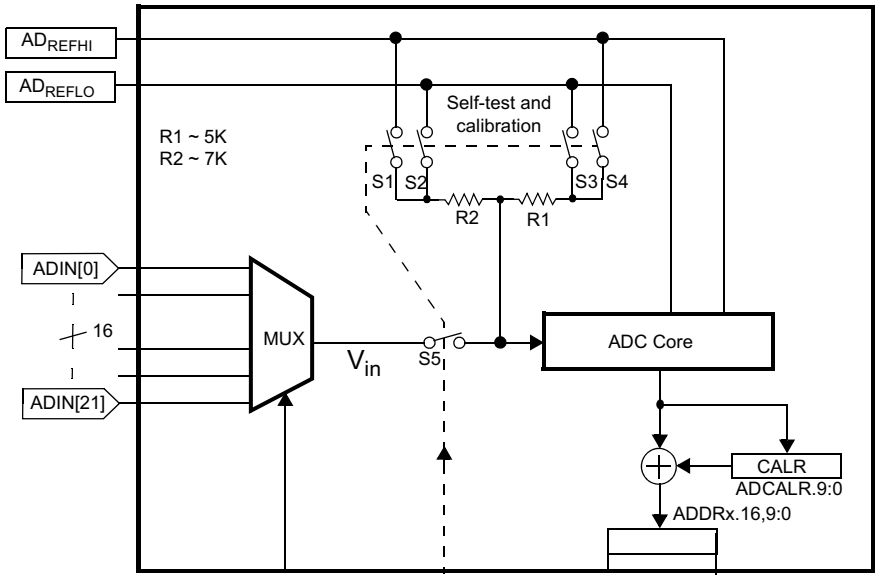
To avoid errors during the calibration operation, self-test mode must *not* be enabled during a calibration sequence. In addition, to ensure accurate results, calibrate the ADC in an environment with minimum noise.

Calibration mode is enabled by setting the CAL_EN bit (ADCALCR.0). The application needs to ensure that no conversion group is being serviced when the calibration mode is enabled.

The input multiplexor gets disabled and only the reference voltage is connected to the ADC core input. Switch S5 of Figure 16-11 is opened. In addition, the digital result issued from a conversion is output from the ADC core to the calibration and offset error correction register, ADCALR. The ADC results' memory is not affected by the calibration conversion.

When calibration mode is disabled, the ADC can be configured for normal conversions.

Figure 16-11. Self-Test and Calibration Logic



16.7.1.1 Calibration Conversion

The calibration conversion also needs to meet the minimum sampling time specification for the ADC. This value is typically 1 us. The Event Group sample time register (ADEVSAMP) is used to specify the number of ADCLK cycles for the calibration conversion.

The BRIDGE_EN and HILO bits (ADCALCR.9:8) control the voltage to the calibration reference device shown in Figure 16-13. The positions of the switches in calibration mode are listed in Table 16-2.

Table 16-2. Calibration Reference Voltages[1]

CAL_EN	BRIDGE_EN	HILO	S1	S2	S3	S4	S5	Reference Voltage
1	0	0	1	0	1	0	0	$(AD_{REFHI} \times R1 + AD_{REFLO} \times R2) / (R1 + R2)$
1	0	1	0	1	0	1	0	$(AD_{REFLO} \times R1 + AD_{REFHI} \times R2) / (R1 + R2)$
1	1	0	0	1	1	0	0	AD_{REFLO}
1	1	1	1	0	0	1	0	AD_{REFHI}
0	X	X	0	0	0	0	1	V_{in}

(1) The state of the switches in this table assumes that self-test mode is not enabled.

When CAL_ST (ADCALCR.16) is set, a calibration conversion is started. The voltage source selected via the BRIDGE_EN and HILO bits is converted once (single conversion mode) and the digital result is returned to the calibration and correction register, ADCALR, where it can be read by the CPU. The CAL_ST bit acts as a flag and must be polled by the CPU. It is held set during the conversion process and automatically clears to indicate the end of the reference voltage conversion.

NOTE: No Interrupt for end of calibration

The ADC does not generate an interrupt to signal the end of the calibration conversion. The application must poll the CAL_ST bit to determine the end of the calibration conversion.

After the CAL_ST bit is set by the application program, it can only be reset by the end of the ongoing conversion generated by the ADC core. If the calibration conversion is interrupted (CAL_EN bit is cleared), the CAL_ST bit is held at 1 until a new calibration conversion has been set and completed. Setting the CAL_ST bit while calibration is disabled (CAL_EN = 0) has no effect; however, in this situation, setting CAL_EN immediately starts a calibration conversion. When the calibration conversion is interrupted by an ADC enable (ADC_EN = 0, CAL_EN = 1, and CAL_ST = 1), a new conversion is automatically restarted as soon as the ADC enable bit is released (ADC_EN = 1).

16.7.1.2 Calibration and Offset Error Correction Sequences

The number of measurements and the source to measure for an ADC calibration are application dependent. The CAL_ST bit must be set for each calibration source to be measured. While calibration mode is enabled, any available calibration sources can be converted according to the BRIDGE_EN and HILO bits (see Table 16-2). The digital results of the calibration measurements should be read from ADCALR by the application after each reference conversion so that a correction value can be computed and written back into ADCALR.

When the application has the necessary calibration data, it should compute the offset error correction value and load it into the calibration and correction register, ADCALR. After the CAL_EN bit is cleared, normal conversion mode restarts, continuing from where it was frozen, but with the addition of self-correction data.

In normal mode, the self-correction system adds the correction value stored in ADCALR to each digital result before it is written to the respective group's FIFO.

The basic calibration routine is as follows:

1. Enable calibration via CAL_EN (ADCALCR.0).
2. Select the voltage source via BRIDGE_EN and HILO (ADCALCR.9:8).
3. Start the conversion with CAL_ST (ADCALCR.16).
4. Wait for CAL_ST to go to 0.
5. Get the results from ADCALR and save to memory.
6. Loop to step 2 until the calibration conversion data is collected for the desired reference voltages.
7. Compute the error correction value using calibration data saved in memory.
8. Load the ADCALR register with the 2s complement of the computed error correction value.
9. Disable calibration mode.

At this point, the ADC can be configured for normal operation, and it corrects each digital result with the error correction value loaded in ADCALR.

NOTE: Prevent ADC Calibration Data From Being Overwritten

In calibration mode, the conversion result is written to ADCALR which overwrites any previous calibration data; therefore, the ADCALR register must be read before a new conversion is started.

For no correction, a value of 0x0000 must be written to ADCALR. In noncalibration mode, the ADCALR register can be read and written. Any value written to ADCALR in normal mode (CAL_EN = 0) is added to each digital result from the ADC core.

16.7.1.3 Mid-Point Calibration

Because of its connections to the ADC's reference voltage (VrefHi, VrefLo), the precision of the calibration reference is voltage independent. On the other hand, the accuracy of the switched bridge resistor (R1 & R2) relies on the manufacturing process deviation. Consequently, the mid-point voltage's accuracy can be affected due to the imperfections in the two resistors (expected mismatch error is around 1.5%).

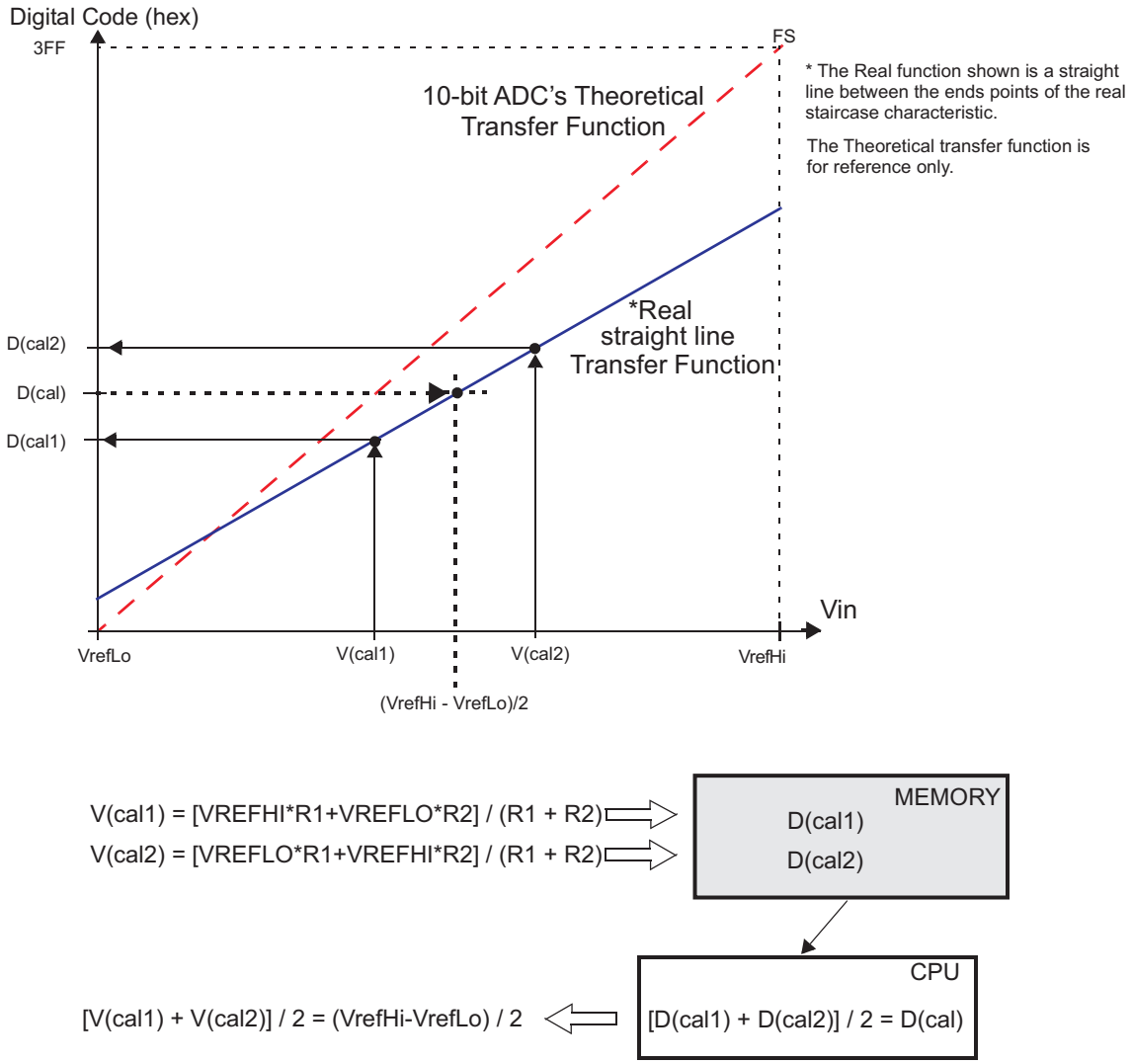
The switched reference voltage device has been specially designed to support a differential measurement of its mid-point voltage. This ensures the accuracy of the mid-point reference, and hence the efficiency of the calibration.

The differential mid-point calibration is software controlled; the algorithm (voltage source measurements and associated calculation) is inserted within the calibration software module included in the application program.

The basic differential mid-point calibration flow is illustrated here after:

1. The application program connects the voltage VrefHi to R1 and VrefLo to R2, (BRIDGE_EN = 0, HILO = 0), launches a conversion of the input voltage V(cal1), and stores the digital result D(cal1) into the memory.
2. Then the application program switches the voltage VrefHi to R2 and VrefLo to R1 (BRIDGE_EN = 0, HILO = 1), converts this new input voltage V(cal2) and again stores the issued digital result D(cal2) into the memory.
3. The actual value of the real middle point is obtained by computing the average of these two results. [D(cal1)+D(cal2)] /2; Figure 16-12 summarizes the mid-point calibration flow.

Figure 16-12. Mid-point Value Calculation

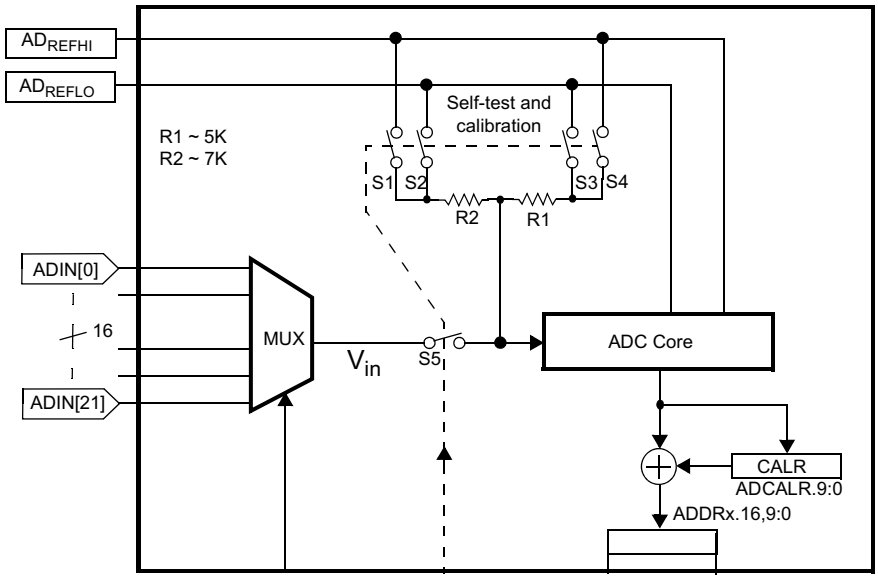


16.7.2 ADC Self-Test Mode

The ADC module supports a self-test mode which can be used to detect an open or a short on the ADC input channels. Self-test mode is enabled by setting the SELF_TEST bit (ADCALCR.24). Any conversion type (continuous or single conversion, freeze enabled or non-freeze enabled, interrupts enabled or disabled) can be performed in this mode.

In normal mode, setting the self-test mode while a conversion sequence is in process can corrupt the current channel conversion results. However, the next channel in the sequence is converted correctly during the additional self-test cycle. The logic associated with both self-test and calibration is shown in Figure 16-13.

Figure 16-13. Self-Test and Calibration Logic



In self-test mode, a test voltage defined by the HILO bit (ADCALCR.8) is provided to the ADC core input through a resistor (see Table 16-3). To change the test source, this bit can be toggled before any single conversion mode request. Changing this bit while a conversion is in progress *can* corrupt the results if the source switches during the acquisition period.

Note that the switch S5 shown in Figure 16-13 is only for the purpose of explaining the self-test sequence. There is no physical switch.

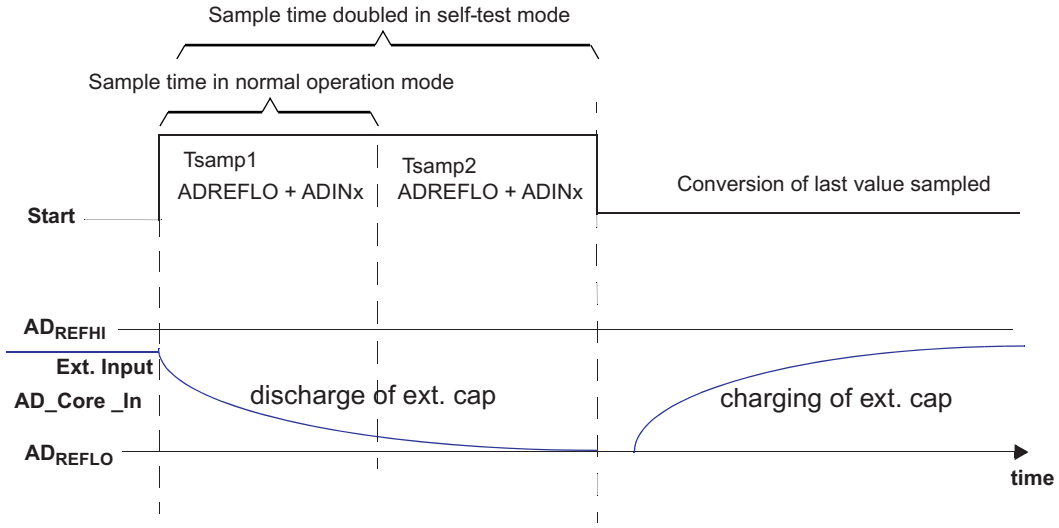
Table 16-3. Self-Test Reference Voltages(1)

SELF_TEST	HILO	S1	S2	S3	S4	S5	Reference Voltage
1	0	0	1	1	0	1	AD_{REFLO} via R1 || R2 connected to V_{in}
1	1	1	0	0	1	1	AD_{REFHI} via R1 || R2 connected to V_{in}
0	X	0	0	0	0	1	V_{in}

(1) Switches refer to Figure 16-13.

Conversions in self-test mode are started just as they are in the normal operating mode (see Section 16.3.6). The conversion starts according to the configuration set in the three mode control registers (ADEVMODECR, ADG1MODECR, ADG2MODECR) and the sampling time control registers (ADEVSAMP, ADG1SAMP, ADG2SAMP). The acquisition time for each conversion in self-test mode is extended to twice the normal configured acquisition time. The selected reference voltage and the input voltage from the ADINx input channel are both connected to the ADC internal sampling capacitor throughout this extended acquisition period. Figure 16-14 shows the self-test mode timing when the ADREFLO is chosen as the reference voltage for the self-test mode conversion. It also assumes an external capacitor connected to the ADC input channel.

Figure 16-14. Timing for Self-Test Mode



16.7.2.1 Use of Self-Test Mode to Determine Open/Short on ADC Input Channels

The following sequence needs to be used to deduce the ADC pin status:

- Convert the channel with self test enabled and with the reference voltage as Vreflo. Store the conversion result, say Vd.
- Convert the channel with self test enabled and with the reference voltage as Vrefhi. Store the conversion result, say Vu.
- Convert the channel with self test disabled. Store the conversion result, say Vn.

The results can be interpreted using Table 16-4.

Table 16-4. Determination of ADC Input Channel Condition

Normal Conversion Result, Vn	Self-test Conversion Result, Vu	Self-test Conversion Result, Vd	Pin Condition
Vn	$Vn < Vu < AD_{REFHI}$	$AD_{REFLO} < Vd < Vn$	Good
AD_{REFHI}	AD_{REFHI}	approx. AD_{REFHI}	Shorted to AD_{REFHI}
AD_{REFLO}	approx. AD_{REFLO}	AD_{REFLO}	Shorted to AD_{REFLO}
Unknown	AD_{REFHI}	AD_{REFLO}	Open

16.7.3 ADC Power-Down Mode

This is an inactive mode in which the clocks to the ADC module are stopped leaving the module in a static state. The clock to the ADC core (ADCLK) is stopped whenever there are no ongoing conversions. This is the clock-gating implementation requirement. Also, the ADC module places the ADC core into the power down mode such that there is minimal current drawn from the ADC operating and reference supplies.

16.7.3.1 Powering Down Just The ADC Core

The ADC core can be individually powered down without stopping the clocks to the ADC module. This can be done by setting the POWERDOWN bit of the ADC Operating Mode Control Register (ADOPMODECR.3). Whenever a conversion is required the POWERDOWN bit must be cleared, and a minimum time $t_{d(PU-ADV)}$, (see the specific device data sheet for actual value) has to be allowed before starting a new conversion. This wait must be implemented in the application software.

16.7.3.2 Enhanced Power-Down Mode

A bit in the ADC operating mode control register, IDLE_PWRDN (ADOPMODECR.4) enables the enhanced power-down mode of the ADC.

Once this bit is set, the ADC module will power down the ADC core whenever there are no more ongoing or pending ADC conversions. The ADC core will be powered down regardless of the state of the POWERDOWN bit (ADOPMODECR.3).

The ADC module releases the ADC core from power down mode as soon as a new conversion is requested. The ADC logic state machine then has to wait for at least $t_{d(PU-ADV)}$ (see the device data sheet for actual value) before starting a new conversion. The IDLE_PWRDN bit will remain set at all times. The logic state machine can use this bit to determine that it needs to wait for a programmable number of VCLK cycles before it allows the input channel to be sampled. This time is configured by the ADC Power Up Delay Control register (ADPWRUPDLYCTRL).

If IDLE_PWRDN is not set, the ADC module does not wait for any additional delay before sampling the input channel and the application software has to take account of this required delay.

16.7.3.3 Managing Clocks to the ADC Module

The clock to the ADC module can be turned off via the appropriate Peripheral Central Resource (PCR) controller PSPWRDNSET register (check the specific device datasheet to identify the register and the bit to be set). If a conversion is ongoing when this bit is set, the ADC module will wait until the current conversion completes before allowing the ADC module clock to be stopped.

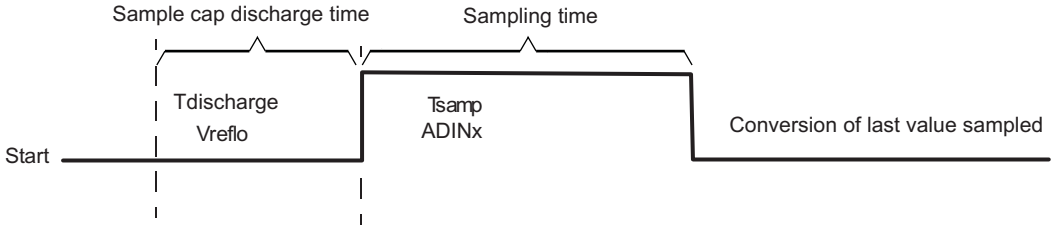
16.7.4 ADC Sample Capacitor Discharge Mode

This mode allows the charge on the ADC core's internal sampling capacitor to be discharged before starting the sampling phase of the next channel.

The ADC Sample Cap Discharge Mode is enabled by setting the SAMP_DIS_EN bit of the group's ADSAMPDISEN register. A discharge period for the sampling capacitor is added before the sampling period for each channel as shown in Figure 16-15. The duration of this discharge period is configurable via the corresponding group's SAMP_DIS_CYC field in the ADSAMPDISEN register. The discharge time is specified in terms of number of ADCLK cycles.

During the sample capacitor discharge period, the V_{REFLO} reference voltage is connected to the input voltage terminal of the ADC core. This allows any charge collected on the sampling capacitor from the previous conversion to be discharged to ground. The V_{REFLO} reference voltage is usually connected to ground.

Figure 16-15. Timing for Sample Capacitor Discharge Mode



16.8 ADC Results' RAM Special Features

The following sections describe some of the special features supported by the ADC module to enhance the results' RAM testability and integrity.

16.8.1 ADC Results' RAM Auto-Initialization

The ADC module allows the application to auto-initialize the ADC results' RAM to all zeros. The application must ensure that the ADC module is not in any of the conversion modes before triggering off the auto-initialization process.

The auto-initialization sequence is as follows:

1. Enable the global hardware memory initialization key by programming a value of Ah to bits [3-0] of the MINITGCR register of the System module.
2. Set the control bit for the ADC results' RAM in the MSINENA System module register. Bit 8 of the MSINENA register is used to control the initialization of the ADC results' RAM. This starts the initialization process. The BUF_INIT_ACTIVE flag in the ADC module ADBNDEND register will get set to reflect that the initialization is ongoing.
3. When the memory initialization is completed, the corresponding status bit in the MINISTAT register will be set. Also, the BUF_INIT_ACTIVE flag will get cleared.

16.8.2 ADC Results' RAM Test Mode

In the defined conversion modes of the ADC, the application can only read from the ADC results' RAM. Only the ADC module is allowed to write to the results' RAM. A special test mode is defined to allow the application to also write into the ADC results' RAM - this mode is the ADC Results' RAM Test Mode. Only 32-bit reads and writes are allowed to the ADC results' RAM in this test mode.

NOTE: Contention on access to ADC Results' RAM

The ADC module cannot handle a contention between the application write to the results' RAM and the ADC writing a conversion result to the results' RAM. The application must ensure that the ADC is not likely to write a new conversion result to the results' RAM when the ADC Results' RAM Test Mode is enabled.

The ADC Results' RAM Test Mode is enabled by setting the RAM_TEST_EN bit in the ADOPMODECR.

16.8.3 ADC Results' RAM Parity

The following shows the ADC Results' RAM parity control registers.

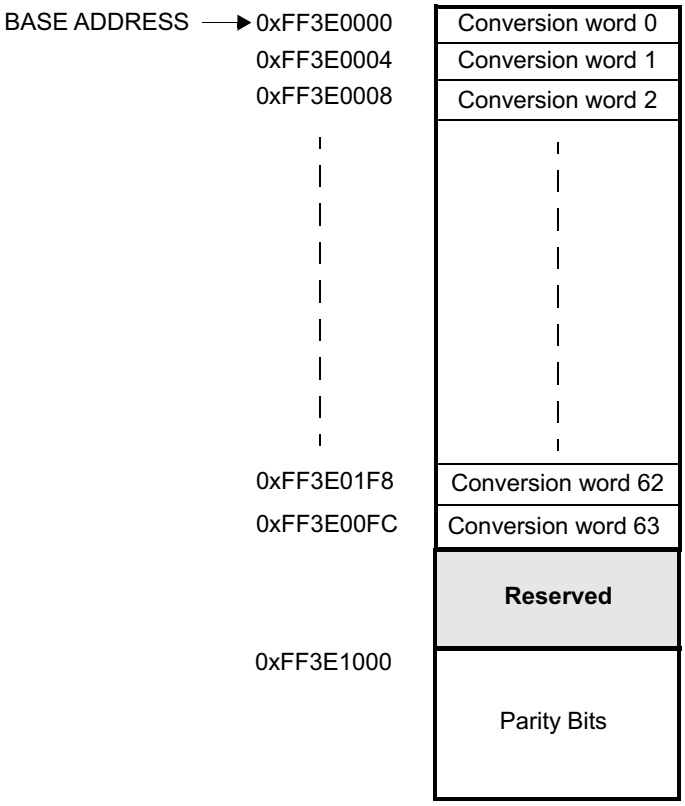
Parity checking is implemented using parity on a per-half word basis for the ADC RAM. That is, there is one parity bit for 16 bits of the ADC RAM. The polarity of the ADC RAM parity is controlled by the DEVCR1 register in the system module (address = FFFF FFDCh). The parity checking is enabled by the ADPARCR register. After reset, the parity checking is disabled and must be enabled if parity protection is required.

During a read access, the parity is calculated based on the data read from the ADC RAM and compared with the good parity value stored in the parity bits. If any word fails the parity check then the ADC generates an error signal hooked up to the Error Signaling Module (ESM). The ADC RAM address which generated the parity error is captured for host system debugging, and is frozen from being updated until it is read by the application.

Testing the Parity Checking Mechanism:

To test the parity checking mechanism itself, the parity RAM is made writable by the CPU in a special test mode. This is done by a control bit, TEST, in the ADPARCR register. Once this bit is set, the parity bits are mapped to an address starting at an address offset of 4KB from the base address of the ADC RAM. See Figure 16-16. The CPU can now manually insert parity errors. Note that the ADC RAM only supports 32-bit accesses.

Figure 16-16. ADC Memory Map in Parity Test Mode



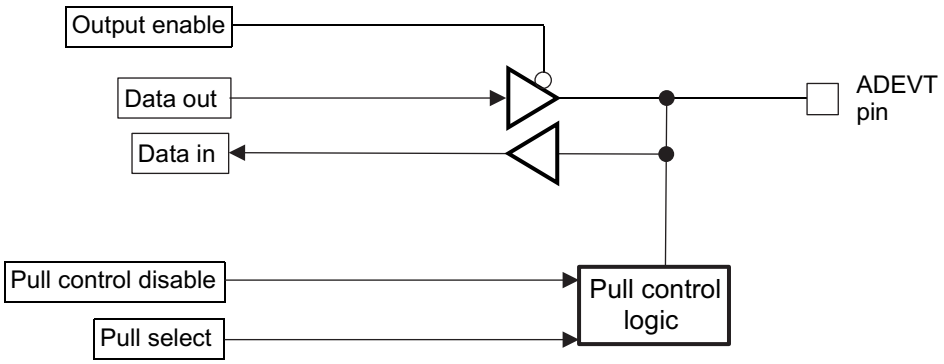
16.9 ADEVT Pin General Purpose I/O Functionality

The ADEVT pin can be configured as general-purpose I/O signals. The following sections describe the different ways in which the application can configure the ADEVT pins.

16.9.1 GPIO Functionality

Figure 16-17 illustrates the GPIO functionality of the ADEVT pin.

Figure 16-17. GPIO Functionality of ADEVT



Once the device power-on reset is released, the ADC module controls the state of the ADEVT pin.

- **Pull control:** The pull control can either be enabled or disabled by default (while system reset is active and after it is released). The actual default state of the pull control is specified in the device datasheet. The application can enable pull control by clearing the PDIS (pull control disable) bit in the ADEVTPDIS register. In this case, if the PSEL (pull select) bit in the ADEVTPSEL register is set, the pin will have a pull-up. If the PSEL bit is cleared, the pin will have a pull-down. If the PDIS bit is set in the control register, there is no pull-up or pull-down on the pin.

NOTE: Pull Behavior when ADEVT is configured as output

If the ADEVT pin is configured as output, then the pulls are disabled automatically. If the pin is configured as input, the pulls are enabled or disabled depending on bit PDIS in the pull disable register ADEVTPDIS.

- **Output buffer:** The ADEVT pin can be driven as an output pin if the ADEVTDIR bit is set in the pin direction control register.
- **Open-Drain Feature:** The open drain output capability is enabled via the ADEVTPDR control register. The ADEVT pin must be also configured to be an output pin for this mode.
  - The output buffer is enabled if a low signal is being driven on to the pin.
  - The output buffer is disabled if a high signal is being driven on to the pin.

16.9.2 Summary

The behavior of the output buffer, and the pull control is summarized in Table 16-5. The input buffer for the ADEVT pin is enabled once the device power-on reset is released.

Table 16-5. Output Buffer and Pull Control Behavior for ADEVT as GPIO Pins

System Reset Active?	Pin Direction (DIR) (1)(2)	Pull Disable (PDIS) (1)(3)	Pull Select (PSEL) (1)(4)	Pull Control	Output Buffer
Yes	X	X	X	Enabled	Disabled
No	0	0	0	Pull down	Disabled
No	0	0	1	Pull up	Disabled
No	0	1	0	Disabled	Disabled
No	0	1	1	Disabled	Disabled
No	1	X	X	Disabled	Enabled

(1) X = Don't care
(2) DIR = 0 for input, 1 for output
(3) PDIS = 0 for enabling pull control, 1 for disabling pull control
(4) PSEL= 0 for pull-down functionality, 1 for pull-up functionality

16.10 ADC Registers

All registers in the ADC module are 32-bit, word-aligned; 8-bit, 16-bit and 32-bit accesses are allowed. The application must ensure that the reserved bits are always written as 0 to ensure software compatibility to future revisions of the module. Table 16-6 shows register address offsets from the base address of the ADC modules. The base address of ADC registers is FFF7 C000h.

Table 16-6. ADC Registers

Offset	Acronym	Register Description	Section
00h	ADRSTCR	ADC Reset Control Register	Section 16.10.1
04h	ADOPMODECR	ADC Operating Mode Control Register	Section 16.10.2
08h	ADCLOCKCR	ADC Clock Control Register	Section 16.10.3
0Ch	ADCALCR	ADC Calibration Mode Control Register	Section 16.10.4
10h	ADEVMODECR	ADC Event Group Operating Mode Control Register	Section 16.10.5
14h	ADG1MODECR	ADC Group1 Operating Mode Control Register	Section 16.10.6
18h	ADG2MODECR	ADC Group2 Operating Mode Control Register	Section 16.10.7
1Ch	ADEVSRC	ADC Trigger Source Select Register	Section 16.10.8
20h	ADG1SRC	ADC Group1 Trigger Source Select Register	Section 16.10.9
24h	ADG2SRC	ADC Group2 Trigger Source Select Register	Section 16.10.10
28h	ADEVINTENA	ADC Event Interrupt Enable Control Register	Section 16.10.11
2Ch	ADG1INTENA	ADC Group1 Interrupt Enable Control Register	Section 16.10.12
30h	ADG2INTENA	ADC Group2 Interrupt Enable Control Register	Section 16.10.13
34h	ADEVINTFLG	ADC Event Group Interrupt Flag Register	Section 16.10.14
38h	ADG1INTFLG	ADC Group1 Interrupt Flag Register	Section 16.10.15
3Ch	ADG2INTFLG	ADC Group2 Interrupt Flag Register	Section 16.10.16
40h	ADEVTHRINTCR	ADC Event Group Threshold Interrupt Control Register	Section 16.10.17
44h	ADG1THRINTCR	ADC Group1 Threshold Interrupt Control Register	Section 16.10.18
48h	ADG2THRINTCR	ADC Group2 Threshold Interrupt Control Register	Section 16.10.19
58h	ADBNDCR	ADC Results Memory Configuration Register	Section 16.10.20
5Ch	ADBNDEND	ADC Results Memory Size Configuration Register	Section 16.10.21
60h	ADEVSAMP	ADC Event Group Sampling Time Configuration Register	Section 16.10.22
64h	ADG1SAMP	ADC Group1 Sampling Time Configuration Register()	Section 16.10.23
68h	ADG2SAMP	ADC Group2 Sampling Time Configuration Register	Section 16.10.24
6Ch	ADEVSR	ADC Event Group Status Register	Section 16.10.25
70h	ADG1SR	ADC Group1 Status Register	Section 16.10.26
74h	ADG2SR	ADC Group2 Status Register	Section 16.10.27
78h	ADEVSEL	ADC Event Group Channel Select Register	Section 16.10.28
7Ch	ADG1SEL	ADC Group1 Channel Select Register	Section 16.10.29
80h	ADG2SEL	ADC Group2 Channel Select Register	Section 16.10.30
84h	ADCALR	ADC Calibration and Error Offset Correction Register	Section 16.10.31
88h	ADSMSTATE	ADC State Machine Status Register	Section 16.10.32
8Ch	ADLASTCONV	ADC Channel Last Conversion Value Register	Section 16.10.33
90h-AFh	ADEVBUFFER	ADC Event Group Results FIFO Register	Section 16.10.34
B0h-CFh	ADG1BUFFER	ADC Group1 Results FIFO Register	Section 16.10.35
D0h-EFh	ADG2BUFFER	ADC Group2 Results FIFO Register	Section 16.10.36
F0h	ADEVEMUBUFFER	ADC Event Group Results Emulation FIFO Register	Section 16.10.37
F4h	ADG1EMUBUFFER	ADC Group1 Results Emulation FIFO Register	Section 16.10.38
F8h	ADG2EMUBUFFER	ADC Group2 Results Emulation FIFO Register	Section 16.10.39
FCh	ADEVTDIR	ADC ADEVT Pin Direction Control Register	Section 16.10.40
100h	ADEVTOUT	ADC ADEVT Pin Output Value Control Register	Section 16.10.41
104h	ADEVTIN	ADC ADEVT Pin Input Value Register	Section 16.10.42

Table 16-6. ADC Registers (continued)

Offset	Acronym	Register Description	Section
108h	ADEVTSET	ADC ADEVT Pin Set Register	Section 16.10.43
10Ch	ADEVTCLR	ADC ADEVT Pin Clear Register	Section 16.10.44
110h	ADEVTPDR	ADC ADEVT Pin Open Drain Enable Register	Section 16.10.45
114h	ADEVTPDIS	ADC ADEVT Pin Pull Control Disable Register	Section 16.10.46
118h	ADEVTPSEL	ADC ADEVT Pin Pull Control Select Register	Section 16.10.47
11Ch	ADEVSAMPDISEN	ADC Event Group Sample Cap Discharge Control Register	Section 16.10.48
120h	ADG1SAMPDISEN	ADC Group1 Sample Cap Discharge Control Register	Section 16.10.49
124h	ADG2SAMPDISEN	ADC Group2 Sample Cap Discharge Control Register	Section 16.10.50
128h-138h	ADMAGINTxCR	ADC Magnitude Compare Interrupt x Control Register	Section 16.10.51
12Ch-13Ch	ADMAGxMASK	ADC Magnitude Compare Interrupt x Mask Register	Section 16.10.52
158h	ADMAGINTENASET	ADC Magnitude Compare Interrupt Enable Set Register	Section 16.10.53
15Ch	ADMAGINTENACLR	ADC Magnitude Compare Interrupt Enable Clear Register	Section 16.10.54
160h	ADMAGINTFLG	ADC Magnitude Compare Interrupt Flag Register	Section 16.10.55
164h	ADMAGINTOFF	ADC Magnitude Compare Interrupt Offset Register	Section 16.10.56
168h	ADEVFIFORESETCR	ADC Event Group FIFO Reset Control Register	Section 16.10.57
16Ch	ADG1FIFORESETCR	ADC Group1 FIFO Reset Control Register	Section 16.10.58
170h	ADG2FIFORESETCR	ADC Group2 FIFO Reset Control Register	Section 16.10.59
174h	ADEVRAMWRADDR	ADC Event Group RAM Write Address Register	Section 16.10.60
178h	ADG1RAMWRADDR	ADC Group1 RAM Write Address Register	Section 16.10.61
17Ch	ADG2RAMWRADDR	ADC Group2 RAM Write Address Register	Section 16.10.62
180h	ADPARCR	ADC Parity Control Register	Section 16.10.63
184h	ADPARADDR	ADC Parity Error Address Register	Section 16.10.64
188h	ADPWRUPDLYCTRL	ADC Power-Up Delay Control Register	Section 16.10.65
190h	ADEVCHNSELMODECTRL	ADC Event Group Channel Selection Mode Control Register	Section 16.10.66
194h	ADG1CHNSELMODECTRL	ADC Group1 Channel Selection Mode Control Register	Section 16.10.67
198h	ADG2CHNSELMODECTRL	ADC Group2 Channel Selection Mode Control Register	Section 16.10.68
19Ch	ADEVCURRCOUNT	ADC Event Group Current Count Register	Section 16.10.69
1A0h	ADEVMAXCOUNT	ADC Event Group Max Count Register	Section 16.10.70
1A4h	ADG1CURRCOUNT	ADC Group1 Current Count Register	Section 16.10.71
1A8h	ADG1MAXCOUNT	ADC Group1 Max Count Register	Section 16.10.72
1ACh	ADG2CURRCOUNT	ADC Group2 Current Count Register	Section 16.10.73
1B0h	ADG2MAXCOUNT	ADC Group2 Max Count Register	Section 16.10.74

16.10.1 ADC Reset Control Register (ADRSTCR)

Figure 16-18 and Table 16-7 describe the ADRSTCR register.

Figure 16-18. ADC Reset Control Register (ADRSTCR) [offset = 00h]

31	1	0
Reserved		RESET
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 16-7. ADC Reset Control Register (ADRSTCR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RESET		This bit is used to reset the ADC internal state machines and control/status registers. This reset state is held until this bit is cleared. Read in all modes, write in privileged mode.
		0	Module is released from the reset state.
		1	All the module's internal state machines and the control/status registers are reset.

16.10.2 ADC Operating Mode Control Register (ADOPMODECR)

Figure 16-19 and Table 16-8 describe the ADOPMODECR register.

Figure 16-19. ADC Operating Mode Control Register (ADOPMODECR) [offset = 04h]

31	30	25	24
10_12_BIT	Reserved		COS
R/W-0	R-0		RW-1

23	21	20	17	16
Reserved		CHN_TEST_EN		RAM_TEST_EN
R-0		R/W-Ah		R/W-0

15	9	8
Reserved		POWERDOWN
R-0		RW-0

7	5	4	3	1	0
Reserved		IDLE_PWRDN	Reserved		ADC_EN
R-0		R/W-0	R-0		RW-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-8. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions

Bit	Field	Value	Description
31	10_12_BIT		This bit controls the resolution of the ADC core. It also affects the size of the conversion results stored in the results' RAM.
			Any operation mode read/write:
		0	The ADC core and digital logic are configured to be in 10-bit resolution. This is the default mode of operation.
		1	The ADC core and digital logic are configured to be in 12-bit resolution.
30-25	Reserved	0	Reads return 0. Writes have no effect.
24	COS		This bit affects *emulation operation only*. It defines whether the ADC core clock (ADCLK) is immediately halted when the emulation system enters suspend mode or if it should continue operating normally.
			Note: If COS = 0 when the ADC module enters the emulation mode, then the accuracy of the conversion results can be affected depending on how long the module stays in the emulation mode.
			Any operation mode read/write:
		0	ADC module halts all ongoing conversions immediately after emulation mode is entered.
		1	ADC module continues all ongoing conversions as per the configurations of the three conversion groups.
23-21	Reserved	0	Reads return 0. Writes have no effect.
20-17	CHN_TEST_EN		Enable the input channels' impedance measurement mode.
			This mode is reserved for use by TI.
			Any operation mode read/write:
		5h	Input impedance measurement mode is enabled.
		Ah	Input impedance measurement mode is disabled.
		All other values	Input impedance measurement mode is disabled.
16	RAM_TEST_EN		Enable the ADC Results' RAM Test Mode.
			Refer to Section 16.8.2 for more details.
			Any operation mode read/write:
		0	ADC RAM Test Mode is disabled. The application cannot write to the ADC RAM.
		1	ADC RAM Test Mode is enabled. The application can directly write to the ADC RAM.
15-9	Reserved	0	Reads return 0. Writes have no effect.

Table 16-8. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions (continued)

Bit	Field	Value	Description
8	POWERDOWN		ADC Power Down. This bit powers down only the ADC core; the digital logic in the sequencer stays active. To release the core from power down mode, this bit must be cleared. If a conversion is ongoing, the ADC module will wait until the current conversion is completed before powering down the ADC core. Also refer to Section 16.10.65, ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL). Any operation mode read/write:
		0	The state of the ADC core is controlled by the IDLE_PWRDN bit, or by a global power down mode entry.
		1	ADC core is in the power-down state.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	IDLE_PWRDN		ADC Power Down When Idle. When this bit is set, the ADC module will automatically power down the ADC core whenever there are no conversions ongoing or pending. This is the enhanced power down mode. Also refer to Section 16.10.65, ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL). Any operation mode read/write:
		0	The ADC stays in the normal operating mode even if no conversions are ongoing or pending. The power down state is entered only by configuring the POWERDOWN bit or via a global power down mode entry.
		1	Enhanced power down mode is enabled.
3-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADC_EN		ADC Enable. This bit must be set to allow the ADC module to be configured to perform any conversions. Any operation mode read/write:
		0	No ADC conversions can occur. The input channel select registers: ADEVSEL, ADG1SEL, and ADG2SEL are held at their reset values.
		1	ADC conversions can now proceed as configured.

16.10.3 ADC Clock Control Register (ADCLOCKCR)

Figure 16-20 and Table 16-9 describe the ADCLOCKCR register.

Figure 16-20. ADC Clock Control Register (ADCLOCKCR) [offset = 08h]

31	5	4	0
Reserved		PS	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-9. ADC Clock Control Register (ADCLOCKCR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	PS	0-1Fh	ADC Clock Prescaler. These bits define the prescaler value for the ADC core clock (ADCLK). The ADCLK is generated by dividing down the input bus clock (VCLK) to the ADC module. **Note:** The supported range for the ADC clock frequency is specified in the device datasheet. The ADC clock prescaler must be configured to meet this datasheet specification. Any operation mode read/write: $t_{C(ADCLK)} = t_{C(VCLK)} \times (PS + 1)$, where $t_{C(ADCLK)}$ is the period of the ADCLK and $t_{C(VCLK)}$ is the period of the VCLK.

16.10.4 ADC Calibration Mode Control Register (ADCALCR)

Figure 16-21 and Table 16-10 describe the ADCALCR register.

Figure 16-21. ADC Calibration Mode Control Register (ADCALCR) [offset = 0Ch]

31	25	24
Reserved		SELF_TEST
R-0		RW-0

23	17	16
Reserved		CAL_ST
R-0		R/S-0

15	10	9	8
Reserved		BRIDGE_EN	HILO
R-0		RW-0	RW-0

7	1	0
Reserved		CAL_EN
R-0		RW-0

LEGEND: R/W = Read/Write; R = Read only; S = Set; -*n* = value after reset

Table 16-10. ADC Calibration Mode Control Register (ADCALCR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	SELF_TEST		ADC Self Test Enable. When this bit is Set, either AD_{REFHI} or AD_{REFLO} is connected through a resistor to the selected input channel. The desired conversion mode is configured in the group mode control registers. For more details on the ADC Self Test Mode, refer to Section 16.7.2.
			Any operation mode read/write:
		0	ADC Self Test mode is disabled.
		1	ADC Self Test mode is enabled.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	CAL_ST		ADC Calibration Conversion Start. Setting the CAL_ST bit while the CA_ EN bit is set starts conversion of the selected reference voltage. The ADC module uses the sample time configured in the Event Group sample time configuration register (ADEVSAMP) for the calibration conversion.
			Any operation mode:
		0	Read: Calibration conversion has completed, or has not yet been started.
			Write: Writing 0 to this bit has no effect.
		1	Read: Calibration conversion is in progress.
			Write: ADC module starts calibration conversion.
15-10	Reserved	0	Reads return 0. Writes have no effect.
9	BRIDGE_EN		Bridge Enable. When set with the HILO bit, BRIDGE_EN allows a reference voltage to be converted in calibration mode. Table 16-2 defines the four different reference voltages that can be selected.
8	HILO		ADC Self Test mode and Calibration Mode Reference Source Selection.
			In the ADC Self Test mode, this bit defines the test voltage to be combined through a resistor with the selected input pin voltage. Refer to Section 16.7.2 for details on the ADC Self Test Mode.
			In the ADC Calibration Mode, this bit defines the reference source polarity. Refer to Section 16.7.1 for details on the ADC Calibration Mode.
			In the ADC module's normal operating mode, this bit has no effect.
7-1	Reserved	0	Reads return 0. Writes have no effect.

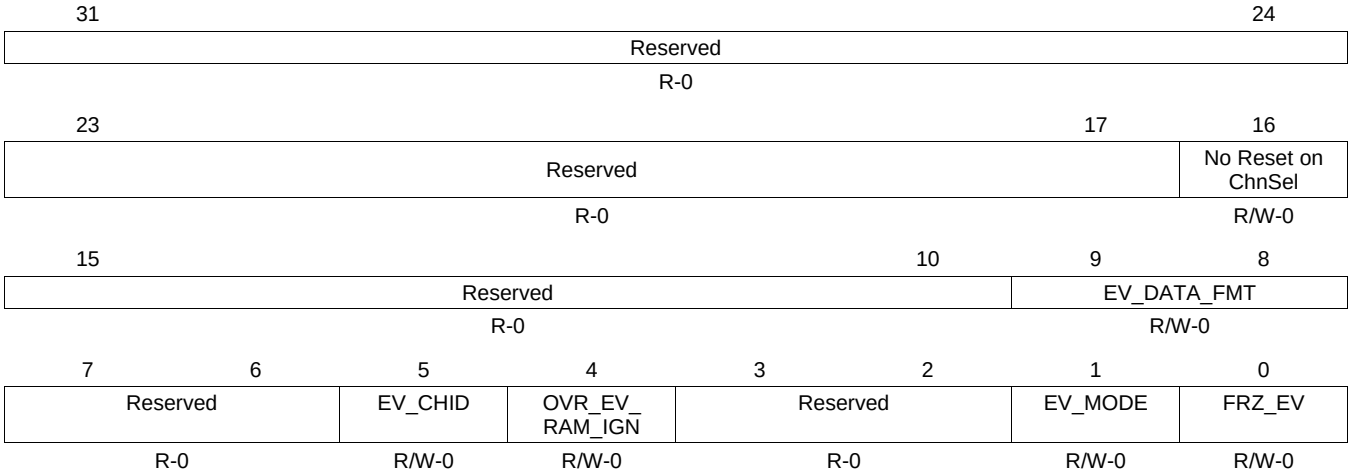
Table 16-10. ADC Calibration Mode Control Register (ADCALCR) Field Descriptions (continued)

Bit	Field	Value	Description
0	CAL_EN		ADC Calibration Enable. When this bit is Set, the input channel multiplexor is disconnected and the calibration reference voltage is connected to the ADC core input. The calibration reference voltage is selected by the combination of the BRIDGE_EN and HILO. The actual conversion of this reference voltage starts when the CAL_ST bit is set. If the CAL_ST bit is already set when the CAL_EN bit is set, then the calibration conversion is immediately started. Refer to Section 16.7.1 for more details on the ADC calibration mode. Any operation mode read/write:
		0	Calibration mode is disabled.
		1	Calibration mode is enabled.

16.10.5 ADC Event Group Operating Mode Control Register (ADEVMODECR)

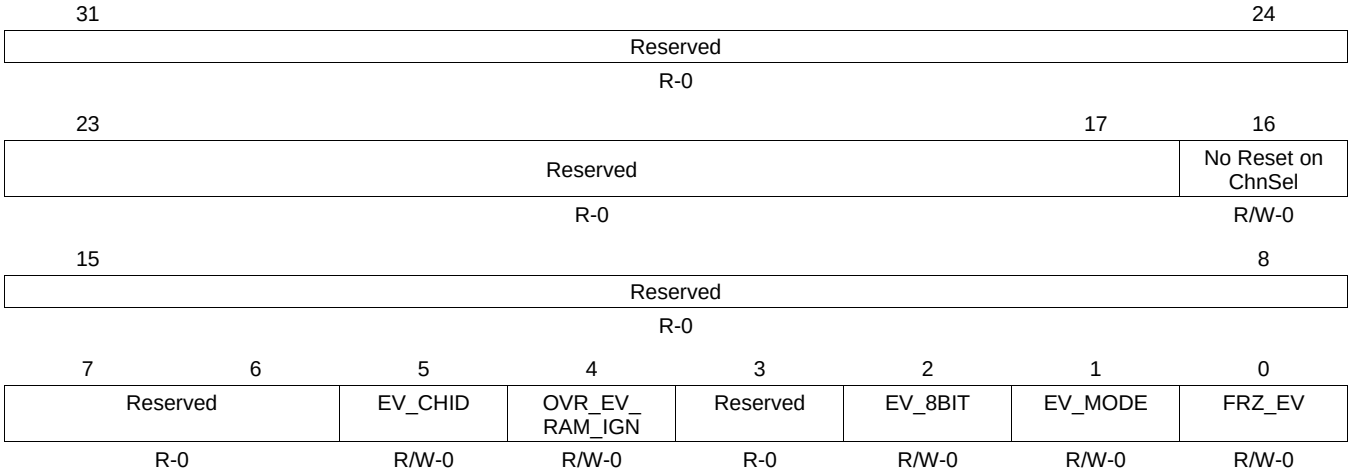
ADC Event Group Operating Mode Control Register (ADEVMODECR) is shown in Figure 16-22 and Figure 16-23, and described in Table 16-11. As shown, the format of the ADEVMODECR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 16-22. 12-bit ADC Event Group Operating Mode Control Register (ADEVMODECR) [offset = 10h]



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 16-23. 10-bit ADC Event Group Operating Mode Control Register (ADEVMODECR) [offset = 10h]



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-11. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
No Reset on ChnSel		No Event Group Results Memory Reset on New Channel Select. This bit determines whether the event group results' RAM is reset whenever a non-zero value is written to the event group channel select register. Any operation mode read/write:
	0	Event group results RAM is reset when a non-zero value is written to event group channel select register, even if event group conversions are completed.
	1	Event group results RAM is not reset when a non-zero value is written to event group channel select register, and event group conversions are completed. If the event group conversions are ongoing (active or frozen), then writing a non-zero value to the event group channel select register will always reset the event group results RAM.
EV_DATA_FMT		Event Group Read Data Format. **Note:** This field is only applicable when the ADC module is configured to be a 12-bit ADC module. This field is reserved when the module is configured as a 10-bit ADC module. This field determines the format in which the conversion results are read out of the Event group results RAM when using the FIFO interface, that is, when reading from the ADEVBUFFER or ADEVEMUBUFFER locations. Any operation mode read/write:
	0	Conversion results are read out in full 12-bit format. This is the default mode.
	1h	Conversion results are read out in 10-bit format. Bits 11-2 of the 12-bit conversion result are returned as the 10-bit conversion result.
	2h	Conversion results are read out in 8-bit format. Bits 11-4 of the 12-bit conversion result are returned as the 8-bit conversion result.
	3h	Reserved. The full 12-bit conversion result is returned if programmed.
EV_CHID		Enable Channel Id for the Event Group conversion results to be read. This bit only affects the "read from FIFO" mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the "read from RAM" mode will return the 5-bit channel id along with the 10-bit conversion result. Any operation mode read/write:
	0	Bits 14-10, the channel id field, of the data read from the Event Group results' FIFO is read as 00000b.
	1	Bits 14-10, the channel id field, of the data read from the Event Group results' FIFO contains the number of the ADC analog input to which the conversion result belongs.
OVR_EV_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Event Group results memory under an overrun condition. Any operation mode read/write:
	0	The ADC cannot overwrite the contents of the Event Group results memory. When an overrun of this memory occurs, the software needs to read out all the contents of this memory before the ADC is able to write a new conversion result for the Event Group.
	1	When an overrun of the Event Group results memory occurs, the ADC proceeds to overwrite the contents with any new conversion results for the Event Group, starting with the first location in this memory.
EV_8BIT		Event Group 8-bit result mode. This bit allows the Event Group conversion results to be read out in an 8-bit format. This bit only applies to the "read from FIFO" mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result. **Note:** This bit is only applicable when the ADC module is configured to be a 10-bit ADC module. This bit is reserved when the module is configured as a 12-bit ADC module. Any operation mode read/write:
	0	The Event Group conversion result is read out as a 10-bit value in the "read from Event Group FIFO" mode.
	1	The Event Group conversion result is read out as an 8-bit value in the "read from Event Group FIFO" mode.

Table 16-11. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions (continued)

Field	Value	Description
EV_MODE		Event Group Conversion Mode. This bit defines whether the input channels selected for conversion in the Event Group are converted only once per trigger, or are continuously converted.
		Any operation mode read/write:
	0	The channels selected for conversion in the Event Group are converted only once when the selected event trigger condition occurs.
	1	The channels selected for conversion in the Event Group are converted continuously when the selected event trigger condition occurs.
FRZ_EV		Event Group Freeze Enable. This bit allows an Event Group conversion sequence to be frozen if a Group1 or a Group2 conversion is requested. The Event Group conversion is kept frozen while the Group1 or Group2 conversion is active, and continues from where it was frozen once the Group1 or Group2 conversions are completed.
		While the Event Group conversion is frozen, the EV_STOP status flag in the ADEVSR register indicates that the Event Group conversions have stopped. This bit gets cleared when the Event Group conversions resume.
		Any operation mode read/write:
	0	Event Group conversions cannot be frozen. All the channels selected for conversion in the Event Group are converted before the ADC can switch over to servicing any other conversion group.
	1	Event Group conversions are frozen whenever there is a request for conversion from Group1 or Group2.

16.10.6 ADC Group1 Operating Mode Control Register (ADG1MODECR)

ADC Group1 Operating Mode Control Register (ADG1MODECR) is shown in Figure 16-24 and Figure 16-25, and described in Table 16-12. As shown, the format of the ADG1MODECR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 16-24. 12-bit ADC Group1 Operating Mode Control Register (ADG1MODECR) [offset = 14h]

31–24	23–17	16	15–10	9–8	7	6	5	4	3	2	1	0
Reserved	Reserved	No Reset on ChnSel	Reserved	G1_DATA_FMT	Reserved		G1_CHID	OVR_G1_RAM_IGN	G1_HW_TRIG	Reserved	G1_MODE	FRZ_G1
R-0	R-0	R/W-0	R-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 16-25. 10-bit ADC Group1 Operating Mode Control Register (ADG1MODECR) [offset = 14h]

31–24	23–17	16	15–8	7	6	5	4	3	2	1	0
Reserved	Reserved	No Reset on ChnSel	Reserved	Reserved		G1_CHID	OVR_G1_RAM_IGN	G1_HW_TRIG	G1_8BIT	G1_MODE	FRZ_G1
R-0	R-0	R/W-0	R-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-12. ADC Group1 Operating Mode Control Register (ADG1MODECR) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
No Reset on ChnSel		No Group1 Results Memory Reset on New Channel Select. This bit determines whether the group1 results' RAM is reset whenever a non-zero value is written to the group1 channel select register. Any operation mode read/write:
	0	Group1 results RAM is reset when a non-zero value is written to group1 channel select register, even if group1 conversions are completed.
	1	Group1 results RAM is not reset when a non-zero value is written to group1 channel select register, and group1 conversions are completed. If the group1 conversions are ongoing (active or frozen), then writing a nonzero value to the group1 channel select register will always reset the group1 results RAM.
G1_DATA_FMT		Group1 Read Data Format. **Note:** This field is only applicable when the ADC module is configured to be a 12-bit ADC module. This field is reserved when the module is configured as a 10-bit ADC module. This field determines the format in which the conversion results are read out of the group1 results RAM when using the FIFO interface, that is, when reading from the ADG1BUFFER or ADG1EMUBUFFER locations. Any operation mode read/write:
	0	Conversion results are read out in full 12-bit format. This is the default mode.
	1h	Conversion results are read out in 10-bit format. Bits 11-2 of the 12-bit conversion result are returned as the 10-bit conversion result.
	2h	Conversion results are read out in 8-bit format. Bits 11-4 of the 12-bit conversion result are returned as the 8-bit conversion result.
	3h	Reserved. The full 12-bit conversion result is returned if programmed.
G1_CHID		Enable Channel Id for the Group1 conversion results to be read. This bit only affects the "read from FIFO" mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the "read from RAM" mode will return the 5-bit channel id along with the 10-bit conversion result. Any operation mode read/write:
	0	Bits 14-10, the channel id field, of the data read from the Group1 results' FIFO is read as 00000b.
	1	Bits 14-10, the channel id field, of the data read from the Group1 results' FIFO contains the number of the ADC analog input to which the conversion result belongs.
OVR_G1_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Group1 results memory under an overrun condition. Any operation mode read/write:
	0	The ADC cannot overwrite the contents of the Group1 results memory. When an overrun of this memory occurs, the software needs to read out all the contents of this memory before the ADC is able to write a new conversion result for the Group1.
	1	When an overrun of the Group1 results memory occurs, the ADC proceeds to overwrite the contents with any new conversion results for the Group1, starting with the first location in this memory.
G1_HW_TRIG		Group1 Hardware Triggered. This bit allows the Group1 to be hardware triggered. The Group1 is software triggered by default. For more details on how to trigger a conversion group, refer to Section 16.3.6. Any operation mode read/write:
	0	The Group1 is software-triggered. A Group1 conversion starts whenever the Group1 channel select register (ADG1SEL) is written with a non-zero value.
	1	The Group1 is hardware-triggered. A Group1 conversion starts whenever the Group1 channel select register has a non-zero value, and the specified hardware trigger occurs. The hardware trigger for the Group1 is specified in the Group1 Trigger Source register (ADG1SRC).

Table 16-12. ADC Group1 Operating Mode Control Register (ADG1MODECR) Field Descriptions (continued)

Field	Value	Description
G1_8BIT		Group1 8-bit result mode. **Note:** This bit is only applicable when the ADC module is configured to be a 10-bit ADC module. This bit is reserved when the module is configured as a 12-bit ADC module This bit allows the Group1 conversion results to be read out in an 8-bit format. This bit only applies to the “read from FIFO” mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result. Any operation mode read/write:
	0	The Group1 conversion result is read out as a 10-bit value in the “read from Group1 FIFO” mode.
	1	The Group1 conversion result is read out as an 8-bit value in the “read from Group1 FIFO” mode.
G1_MODE		Group1 Conversion Mode. This bit defines whether the input channels selected for conversion in the Group1 are converted only once, or are continuously converted. Any operation mode read/write:
	0	The channels selected for conversion in the Group1 are converted only once.
	1	The channels selected for conversion in the Group1 are converted continuously.
FRZ_G1		Group1 Freeze Enable. This bit allows a Group1 conversion sequence to be frozen if an Event Group or a Group2 conversion is requested. The Group1 conversion is kept frozen while the Event Group or Group2 conversion is active, and continues from where it was frozen once the Event Group or Group2 conversions are completed. While the Group1 conversion is frozen, the G1_STOP status flag in the ADG1SR register indicates that the Group1 conversions have stopped. This bit gets cleared when the Group1 conversions resume. Any operation mode read/write:
	0	Group1 conversions cannot be frozen. All the channels selected for conversion in the Group1 are converted before the ADC can switch over to servicing any other conversion group.
	1	Group1 conversions are frozen whenever there is a request for conversion from Event Group or Group2.

16.10.7 ADC Group2 Operating Mode Control Register (ADG2MODECR)

ADC Group2 Operating Mode Control Register (ADG2MODECR) is shown in Figure 16-26 and Figure 16-27, described in Table 16-13. As shown, the format of the ADG2MODECR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 16-26. 12-bit ADC Group2 Operating Mode Control Register (ADG2MODECR) [offset = 18h]

31–24	23–16	15–10	9–8
Reserved	Reserved	Reserved	G2_DATA_FMT
R-0	R-0	R-0	R/W-0

7–6	5	4	3	2	1	0
Reserved	G2_CHID	OVR_G2_RAM_IGN	G2_HW_TRIG	Reserved	G2_MODE	FRZ_G2
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

16
No Reset on ChnSel
R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 16-27. 10-bit ADC Group2 Operating Mode Control Register (ADG2MODECR) [offset = 18h]

31–24	23–16	15–8
Reserved	Reserved	Reserved
R-0	R-0	R-0

7–6	5	4	3	2	1	0
Reserved	G2_CHID	OVR_G2_RAM_IGN	G2_HW_TRIG	G2_8BIT	G2_MODE	FRZ_G2
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

16
No Reset on ChnSel
R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-13. ADC Group 2 Operating Mode Control Register (ADG2MODECR) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
No Reset on ChnSel		No Group2 Results Memory Reset on New Channel Select. This bit determines whether the group2 results' RAM is reset whenever a non-zero value is written to the group2 channel select register. Any operation mode read/write:
	0	Group2 results RAM is reset when a non-zero value is written to group2 channel select register, even if group2 conversions are completed.
	1	Group2 results RAM is not reset when a non-zero value is written to group2 channel select register, and group2 conversions are completed. If the group2 conversions are ongoing (active or frozen), then writing a nonzero value to the group2 channel select register will always reset the group2 results RAM.
G2_DATA_FMT		Group2 Read Data Format. **Note:** This field is only applicable when the ADC module is configured to be a 12-bit ADC module. This field is reserved when the module is configured as a 10-bit ADC module. This field determines the format in which the conversion results are read out of the group1 results RAM when using the FIFO interface, that is, when reading from the ADG2BUFFER or ADG2EMUBUFFER locations. Any operation mode read/write:
	0	Conversion results are read out in full 12-bit format. This is the default mode.
	1h	Conversion results are read out in 10-bit format. Bits 11–2 of the 12-bit conversion result are returned as the 10-bit conversion result.
	2h	Conversion results are read out in 8-bit format. Bits 11–4 of the 12-bit conversion result are returned as the 8-bit conversion result.
	3h	Reserved. The full 12-bit conversion result is returned if programmed.
G2_CHID		Enable Channel Id for the Group2 conversion results to be read. This bit only affects the "read from FIFO" mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the "read from RAM" mode will return the 5-bit channel id along with the 10-bit conversion result. Any operation mode read/write:
	0	Bits 14-10, the channel id field, of the data read from the Group2 results' FIFO is read as 00000b.
	1	Bits 14-10, the channel id field, of the data read from the Group2 results' FIFO contains the number of the ADC analog input to which the conversion result belongs.
OVR_G2_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Group2 results memory under an overrun condition. Any operation mode read/write:
	0	The ADC cannot overwrite the contents of the Group2 results memory. When an overrun of this memory occurs, the software needs to read out all the contents of this memory before the ADC is able to write a new conversion result for the Group2.
	1	When an overrun of the Group2 results memory occurs, the ADC proceeds to overwrite the contents with any new conversion results for the Group2, starting with the first location in this memory.
G2_HW_TRIG		Group2 Hardware Triggered. This bit allows the Group2 to be hardware triggered. The Group2 is software triggered by default. For more details on how to trigger a conversion group, refer to Section 16.3.6. Any operation mode read/write:
	0	The Group2 is software-triggered. A Group2 conversion starts whenever the Group2 channel select register (ADG2SEL) is written with a non-zero value.
	1	The Group2 is hardware-triggered. A Group2 conversion starts whenever the Group2 channel select register has a non-zero value, and the specified hardware trigger occurs. The hardware trigger for the Group2 is specified in the Group2 Trigger Source register (ADG2SRC).

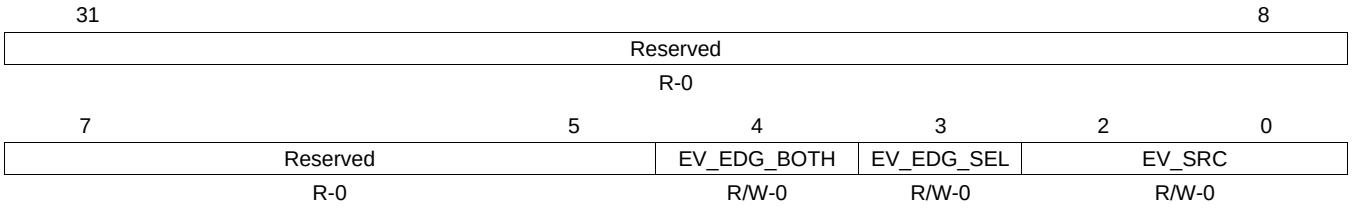
Table 16-13. ADC Group 2 Operating Mode Control Register (ADG2MODECR) Field Descriptions (continued)

Field	Value	Description
G2_8BIT		Group2 8-bit result mode. **Note:** This bit is only applicable when the ADC module is configured to be a 10-bit ADC module. This bit is reserved when the module is configured as a 12-bit ADC module. This bit allows the Group2 conversion results to be read out in an 8-bit format. This bit only applies to the “read from FIFO” mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result. Any operation mode read/write:
	0	The Group2 conversion result is read out as a 10-bit value in the “read from Group2 FIFO” mode.
	1	The Group2 conversion result is read out as an 8-bit value in the “read from Group2 FIFO” mode.
G2_MODE		Group2 Conversion Mode. This bit defines whether the input channels selected for conversion in the Group2 are converted only once, or are continuously converted. Any operation mode read/write:
	0	The channels selected for conversion in the Group2 are converted only once.
	1	The channels selected for conversion in the Group2 are converted continuously.
FRZ_G2		Group2 Freeze Enable. This bit allows a Group2 conversion sequence to be frozen if an Event Group or a Group1 conversion is requested. The Group2 conversion is kept frozen while the Event Group or Group1 conversion is active, and continues from where it was frozen once the Event Group or Group1 conversions are completed. While the Group2 conversion is frozen, the G2_STOP status flag in the ADG2SR register indicates that the Group2 conversions have stopped. This bit gets cleared when the Group2 conversions resume. Any operation mode read/write:
	0	Group2 conversions cannot be frozen. All the channels selected for conversion in the Group2 are converted before the ADC can switch over to servicing any other conversion group.
	1	Group2 conversions are frozen whenever there is a request for conversion from Event Group or Group1.

16.10.8 ADC Event Group Trigger Source Select Register (ADEVSRC)

ADC Event Group Trigger Source Select Register (ADEVSRC) is shown in Figure 16-28 and described in Table 16-14.

Figure 16-28. ADC Event Group Trigger Source Select Register (ADEVSRC) [offset = 1Ch]



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

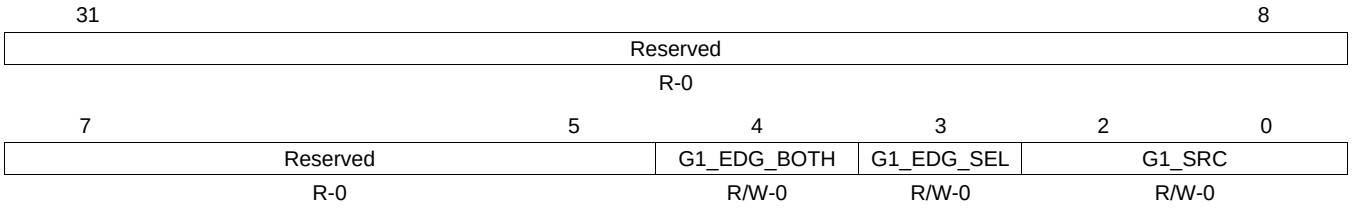
Table 16-14. ADC Event Group Trigger Source Select Register (ADEVSRC) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4	EV_EDG_BOTH		EV_Group Trigger Edge Polarity Select. This bit configures the event group to be triggered on both rising and falling edge detected on the selected trigger source.
			Any operation mode read/write:
		0	The conversion is triggered only upon detecting an edge defined by the EV_EDGE_SEL bit.
		1	The conversion is triggered upon detecting either a rising or falling edge.
3	EV_EDG_SEL		Event Group Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Event Group conversion.
			Any operation mode read/write:
		0	A high-to-low transition on the selected source will trigger the Event Group conversion.
		1	A low-to-high transition on the selected source will trigger the Event Group conversion.
2-0	EV_SRC		Event Group Trigger Source.
			Any operation mode read/write:
		0-7h	The ADC module allows a trigger source to be selected for the Event Group from up to eight options. These options are device-specific and the device specification must be referred to identify the actual trigger sources.

16.10.9 ADC Group1 Trigger Source Select Register (ADG1SRC)

ADC Group1 Trigger Source Select Register (ADG1SRC) is shown in Figure 16-29 and described in Table 16-15.

Figure 16-29. ADC Group1 Trigger Source Select Register (ADG1SRC) [offset = 20h]



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

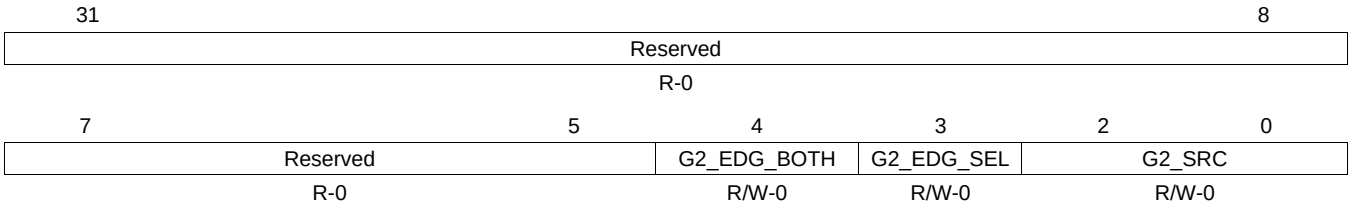
Table 16-15. ADC Group1 Trigger Source Select Register (ADG1SRC) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4	G1_EDG_BOTH		Group1 Trigger Edge Polarity Select. This bit configures the group1 to be triggered on both rising and falling edge detected on the selected trigger source.
			Any operation mode read/write:
		0	The conversion is triggered only upon detecting an edge defined by the G1_EDGE_SEL bit.
		1	The conversion is triggered upon detecting either a rising or falling edge.
3	G1_EDG_SEL		Group1 Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Group1 conversion.
			Any operation mode read/write:
		0	A high-to-low transition on the selected source will trigger the Group1 conversion.
		1	A low-to-high transition on the selected source will trigger the Group1 conversion.
2-0	G1_SRC		Group1 Trigger Source.
			Any operation mode read/write:
		0-7h	The ADC module allows a trigger source to be selected for the Group1 from up to eight options. These options are device-specific and the device specification must be referred to identify the actual trigger sources.

16.10.10 ADC Group2 Trigger Source Select Register (ADG2SRC)

ADC Group2 Trigger Source Select Register (ADG2SRC) is shown in Figure 16-30 and described in Table 16-16.

Figure 16-30. ADC Group2 Trigger Source Select Register (ADG2SRC) [offset = 24h]



LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-16. ADC Group2 Trigger Source Select Register (ADG2SRC) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4	G2_EDG_BOTH		Group2 Trigger Edge Polarity Select. This bit configures the group2 to be triggered on both rising and falling edge detected on the selected trigger source.
			Any operation mode read/write:
		0	The conversion is triggered only upon detecting an edge defined by the G2_EDGE_SEL bit.
		1	The conversion is triggered upon detecting either a rising or falling edge.
3	G2_EDG_SEL		Group2 Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Group2 conversion.
			Any operation mode read/write:
		0	A high-to-low transition on the selected source will trigger the Group2 conversion.
		1	A low-to-high transition on the selected source will trigger the Group2 conversion.
2-0	G2_SRC		Group2 Trigger Source.
			Any operation mode read/write:
		0-7h	The ADC module allows a trigger source to be selected for the Group2 from up to eight options. These options are device-specific and the device specification must be referred to identify the actual trigger sources.

16.10.11 ADC Event Interrupt Enable Control Register (ADEVINTENA)

ADC Event Group Interrupt Enable Control Register (ADEVINTENA) is shown in Figure 16-31 and described in Table 16-17.

Figure 16-31. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) [offset = 28h]

31	8
Reserved	

7	4	3	2	1	0
Reserved		EV_END_INT_EN	Reserved	EV_OVR_INT_EN	EV_THR_INT_EN
R-0		R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-17. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	EV_END_INT_EN		Event Group Conversion End Interrupt Enable. Refer to Section 16.5.1 for more details on the conversion end interrupts.
			Any operation mode read/write:
		0	No interrupt is generated when conversion of all the channels selected for conversion in the Event Group is done.
		1	An Event Group conversion end interrupt is generated when conversion of all the channels selected for conversion in the Event Group is done.
2	Reserved	0	Reads return 0. Writes have no effect.
1	EV_OVR_INT_EN		Event Group Memory Overrun Interrupt Enable. A memory overrun occurs when the ADC tries to write a new conversion result to the Event Group results memory that is already full. For more details on the overrun interrupts, refer to Section 16.5.3.
			Any operation mode read/write:
		0	No interrupt is generated if an Event Group memory overrun occurs.
		1	An Event Group memory overrun interrupt is generated if an Event Group memory overrun condition occurs.
0	EV_THR_INT_EN		Event Group Threshold Interrupt Enable. An Event Group threshold interrupt occurs when the programmed Event Group threshold counter counts down to zero. Refer to Section 16.5.2 for more details.
			Any operation mode read/write:
		0	No interrupt is generated if the Event Group threshold counter reaches zero.
		1	An Event Group threshold interrupt is generated if the Event Group threshold counter reaches zero.

16.10.12 ADC Group1 Interrupt Enable Control Register (ADG1INTENA)

ADC Group1 Interrupt Enable Control Register (ADG1INTENA) is shown in Figure 16-32 and described in Table 16-18.

Figure 16-32. ADC Group1 Interrupt Enable Control Register (ADG1INTENA) [offset = 2Ch]

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		G1_END_INT_EN	Reserved	G1_OVR_INT_EN	G1_THR_INT_EN
R-0		R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-18. ADC Group1 Interrupt Enable Control Register (ADG1INTENA) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	G1_END_INT_EN		Group1 Conversion End Interrupt Enable. Refer to Section 16.5.1 for more details on the conversion end interrupts.
			Any operation mode read/write:
		0	No interrupt is generated when conversion of all the channels selected for conversion in the Group1 is done.
		1	A Group1 conversion end interrupt is generated when conversion of all the channels selected for conversion in the Group1 is done.
2	Reserved	0	Reads return 0. Writes have no effect.
1	G1_OVR_INT_EN		Group1 Memory Overrun Interrupt Enable. A memory overrun occurs when the ADC tries to write a new conversion result to the Group1 results memory that is already full. For more details on the overrun interrupts, refer to Section 16.5.3.
			Any operation mode read/write:
		0	No interrupt is generated if a Group1 memory overrun occurs.
		1	A Group1 memory overrun interrupt is generated if a Group1 memory overrun condition occurs.
0	G1_THR_INT_EN		Group1 Threshold Interrupt Enable. A Group1 threshold interrupt occurs when the programmed Group1 threshold counter counts down to zero. Refer to Section 16.5.2 for more details.
			Any operation mode read/write:
		0	No interrupt is generated if the Group1 threshold counter reaches zero.
		1	A Group1 threshold interrupt is generated if the Group1 threshold counter reaches zero.

16.10.13 ADC Group2 Interrupt Enable Control Register (ADG2INTENA)

ADC Group2 Interrupt Enable Control Register (ADG2INTENA) is shown in Figure 16-33 and described in Table 16-19.

Figure 16-33. ADC Group2 Interrupt Enable Control Register (ADG2INTENA) [offset = 30h]

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		G2_END_INT_EN	Reserved	G2_OVR_INT_EN	G2_THR_INT_EN
R-0		R/W-0	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-19. ADC Group2 Interrupt Enable Control Register (ADG2INTENA) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	G2_END_INT_EN		Group2 Conversion End Interrupt Enable. Refer to Section 16.5.1 for more details on the conversion end interrupts.
			Any operation mode read/write:
		0	No interrupt is generated when conversion of all the channels selected for conversion in the Group2 is done.
		1	A Group2 conversion end interrupt is generated when conversion of all the channels selected for conversion in the Group2 is done.
2	Reserved	0	Reads return 0. Writes have no effect.
1	G2_OVR_INT_EN		Group2 Memory Overrun Interrupt Enable. A memory overrun occurs when the ADC tries to write a new conversion result to the Group2 results memory that is already full. For more details on the overrun interrupts, refer to Section 16.5.3.
			Any operation mode read/write:
		0	No interrupt is generated if a Group2 memory overrun occurs.
		1	A Group2 memory overrun interrupt is generated if a Group2 memory overrun condition occurs.
0	G2_THR_INT_EN		Group2 Threshold Interrupt Enable. A Group2 threshold interrupt occurs when the programmed Group2 threshold counter counts down to zero. Refer to Section 16.5.2 for more details.
			Any operation mode read/write:
		0	No interrupt is generated if the Group2 threshold counter reaches zero.
		1	A Group2 threshold interrupt is generated if the Group2 threshold counter reaches zero.

16.10.14 ADC Event Group Interrupt Flag Register (ADEVINTFLG)

ADC Event Group Interrupt Enable Control Register (ADEVINTENA) is shown in Figure 16-34 and described in Table 16-20.

Figure 16-34. ADC Event Group Interrupt Flag Register (ADEVINTFLG) [offset = 34h]

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		EV_END	EV_MEM_ EMPTY	EV_MEM_ OVERRUN	EV_THR_ INT_FLG
R-0		R/W1C-0	R-1	R-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 16-20. ADC Event Group Interrupt Flag Register (ADEVINTFLG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	EV_END		Event Group Conversion End. This bit will be set only if the Event Group conversions are configured to be in the single-conversion mode. Any operation mode read: 0 All the channels selected for conversion in the Event Group have not yet been converted. 1 All the channels selected for conversion in the Event Group have been converted. An Event Group conversion end interrupt is generated, if enabled, when this bit gets set. This bit can be cleared by any one of the following ways: • By writing a 1 to this bit • By writing a 1 to the Event Group status register bit 0 (EV_END) • By reading one conversion result from the Event Group results' memory in the "read from FIFO" mode • By writing a new set of channels to the Event Group channel select register
2	EV_MEM_EMPTY		Event Group Results Memory Empty. This is a read-only bit; writes have no effect. It is not a source of an interrupt from the ADC module. Any operation mode read: 0 The Event Group results memory is not empty. 1 The Event Group results memory is empty.
1	EV_MEM_OVERRUN		Event Group Memory Overrun. This is a read-only bit; writes have no effect. Any operation mode read: 0 Event Group results memory has not overrun. 1 Event Group results memory has overrun.
0	EV_THR_INT_FLG		Event Group Threshold Interrupt Flag. Any operation mode read: 0 The number of conversions completed for the Event Group is smaller than the threshold programmed in the Event Group interrupt threshold register. 1 The number of conversions completed for the Event Group is equal to or greater than the threshold programmed in the Event Group interrupt threshold register. This bit can be cleared by writing a 1; writing a 0 has no effect.

16.10.15 ADC Group1 Interrupt Flag Register (ADG1INTFLG)

ADC Group1 Interrupt Flag Register (ADG1INTFLG) is shown in Figure 16-35 and described in Table 16-21.

Figure 16-35. ADC Group1 Interrupt Flag Register (ADG1INTFLG) [offset = 38h]

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		G1_END	G1_MEM_ EMPTY	G1_MEM_ OVERRUN	G1_THR_ INT_FLG
R-0		R/W1C-0	R-1	R-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 16-21. ADC Group1 Interrupt Flag Register (ADG1INTFLG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	G1_END		Group1 Conversion End. This bit will be set only if the Group1 conversions are configured to be in the single-conversion mode. Any operation mode read: 0 All the channels selected for conversion in the Group1 have not yet been converted. 1 All the channels selected for conversion in the Group1 have been converted. A Group1 conversion end interrupt is generated, if enabled, when this bit gets set. This bit can be cleared by any one of the following ways: • By writing a 1 to this bit • By writing a 1 to the Group1 status register bit 0 (G1_END) • By reading one conversion result from the Group1 results' memory in the "read from FIFO" mode • By writing a new set of channels to the Group1 channel select register
2	G1_MEM_EMPTY		Group1 Results Memory Empty. This is a read-only bit; writes have no effect. It is not a source of an interrupt from the ADC module. Any operation mode read: 0 The Group1 results memory is not empty. 1 The Group1 results memory is empty.
1	G1_MEM_OVERRUN		Group1 Memory Overrun. This is a read-only bit; writes have no effect. Any operation mode read: 0 Group1 results memory has not overrun. 1 Group1 results memory has overrun.
0	G1_THR_INT_FLG		Group1 Threshold Interrupt Flag. Any operation mode read: 0 The number of conversions completed for the Group1 is smaller than the threshold programmed in the Group1 interrupt threshold register. 1 The number of conversions completed for the Group1 is equal to or greater than the threshold programmed in the Group1 interrupt threshold register. This bit can be cleared by writing a 1; writing a 0 has no effect.

16.10.16 ADC Group2 Interrupt Flag Register (ADG2INTFLG)

ADC Group2 Interrupt Flag Register (ADG2INTFLG) is shown in Figure 16-36 and described in Table 16-22.

Figure 16-36. ADC Group2 Interrupt Flag Register (ADG2INTFLG) [offset = 3Ch]

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		G2_END	G2_MEM_ EMPTY	G2_MEM_ OVERRUN	G2_THR_ INT_FLG
R-0		R/W1C-0	R-1	R-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 16-22. ADC Group2 Interrupt Flag Register (ADG2INTFLG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	G2_END		Group2 Conversion End. This bit will be set only if the Group2 conversions are configured to be in the single-conversion mode. Any operation mode read: 0 All the channels selected for conversion in the Group2 have not yet been converted. 1 All the channels selected for conversion in the Group2 have been converted. A Group2 conversion end interrupt is generated, if enabled, when this bit gets set. This bit can be cleared by any one of the following ways: • By writing a 1 to this bit • By writing a 1 to the Group2 status register bit 0 (G2_END) • By reading one conversion result from the Group2 results' memory in the "read from FIFO" mode • By writing a new set of channels to the Group2 channel select register
2	G2_MEM_EMPTY		Group2 Results Memory Empty. This is a read-only bit; writes have no effect. It is not a source of an interrupt from the ADC module. Any operation mode read: 0 The Group2 results memory is not empty. 1 The Group2 results memory is empty.
1	G2_MEM_OVERRUN		Group2 Memory Overrun. This is a read-only bit; writes have no effect. Any operation mode read: 0 Group2 results memory has not overrun. 1 Group2 results memory has overrun.
0	G2_THR_INT_FLG		Group2 Threshold Interrupt Flag. Any operation mode read: 0 The number of conversions completed for the Group2 is smaller than the threshold programmed in the Group2 interrupt threshold register. 1 The number of conversions completed for the Group2 is equal to or greater than the threshold programmed in the Group2 interrupt threshold register. This bit can be cleared by writing a 1; writing a 0 has no effect.

16.10.17 ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR)

ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) is shown in Figure 16-37 and described in Table 16-23.

Figure 16-37. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) [offset = 40h]

31	16	15	9	8	0
Reserved		Sign Extension		EV_THR	
R-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-23. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-9	Sign Extension		These bits always read the same as the bit 8 of this register.
8-0	EV_THR		Event Group Threshold Counter. Before ADC conversions begin on the Event Group, this field is initialized to the number of conversion results that the Event Group memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Event Group results' memory. The counter increments for each read of a conversion result from the Event Group results' memory in the "read from FIFO" mode. The threshold counter is not affected for a direct read from the Event Group results' memory. Also, a simultaneous ADC write and a CPU read from the Event Group FIFO will leave the threshold counter unchanged. In case of an Event Group Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Event Group threshold counter is not decremented. Refer to Section 16.5.2 for more details on the threshold interrupts.

16.10.18 ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR)

ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) is shown in Figure 16-38 and described in Table 16-24.

Figure 16-38. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) [offset = 44h]

31	16	15	9	8	0
Reserved		Sign Extension		G1_THR	
R-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-24. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-9	Sign Extension		These bits always read the same as the bit 8 of this register.
8-0	G1_THR		Group1 Threshold Counter. Before ADC conversions begin on the Group1, this field is initialized to the number of conversion results that the Group1 memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Group1 results' memory. The counter increments for each read of a conversion result from the Group1 results' memory in the "read from FIFO" mode. The threshold counter is not affected for a direct read from the group1 results' memory. Also, a simultaneous ADC write and a CPU read from the Group1 FIFO will leave the threshold counter unchanged. In case of an Group1 Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Group1 threshold counter is not decremented. Refer to Section 16.5.2 for more details on the threshold interrupts.

16.10.19 ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR)

The ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) is shown in Figure 16-39 and described in Table 16-25.

Figure 16-39. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) [offset = 48h]

31 16	15 9	8 0
Reserved	Sign Extension	G2_THR
R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-25. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-9	Sign Extension		These bits always read the same as the bit 8 of this register.
8-0	G2_THR		Group2 Threshold Counter. Before ADC conversions begin on the Group2, this field is initialized to the number of conversion results that the Group2 memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Group2 results' memory. The counter increments for each read of a conversion result from the Group2 results' memory in the "read from FIFO" mode. The threshold counter is not affected for a direct read from the group2 results' memory. Also, a simultaneous ADC write and a CPU read from the Group2 FIFO will leave the threshold counter unchanged. In case of an Group2 Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Group2 threshold counter is not decremented. Refer to Section 16.5.2 for more details on the threshold interrupts.

16.10.20 ADC Results Memory Configuration Register (ADBNDCR)

ADC Results Memory Configuration Register (ADBNDCR) is shown in Figure 16-40 and described in Table 16-26.

Refer to Section 16.3.8 for further details on how the conversion results are stored in the ADC results' RAM.

Figure 16-40. ADC Results Memory Configuration Register (ADBNDCR) [offset = 58h]

31	25	24	16
Reserved		BNDA	
R-0		R/W-0	

15	9	8	0
Reserved		BNDB	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-26. ADC Results Memory Configuration Register (ADBNDCR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24-16	BNDA		Buffer Boundary A. These bits determine the memory available for the Event Group conversion results. The memory available is specified in terms of pairs of result buffers.
			Any operation mode read/write:
		0	Event Group conversions are not required. If Event Group conversions are performed with the BNDA value of zero, then the Event Group memory size will default to 1024 words. For proper usage of the ADC results memory, configure the BNDA value to be non-zero and lower than the BNDB value.
		0-1FFh	A total of (2 × BNDA) buffers are available in the ADC results memory for storing Event Group conversion results.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	BNDB		Buffer Boundary B. These bits specify the number of buffers allocated for the Event Group plus the number of buffers allocated for the Group1. The number of buffer pairs allocated for storing Group1 conversion results can be determined by subtracting BNDA from BNDB. As a result, BNDB must always be specified as greater than or equal to BNDA.
			Any operation mode read/write:
		0	Event Group as well as Group1 conversions are not required.
		0-1FFh	A total of 2 × (BNDB - BNDA) buffers are available in the ADC results memory for storing Group1 conversion results.

16.10.21 ADC Results Memory Size Configuration Register (ADBNDEND)

ADC Results Memory Size Configuration Register (ADBNDEND) is shown in Figure 16-41 and described in Table 16-27.

Figure 16-41. ADC Results Memory Size Configuration Register (ADBNDEND) [offset = 5Ch]

31	17	16
Reserved		BUF_INIT_ACTIVE
R-0		R-0

15	3	2	0
Reserved		BNDEND	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-27. ADC Results Memory Size Configuration Register (ADBNDEND) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUF_INIT_ACTIVE		ADC Results Memory Auto-initialization Status. Any operation mode read/write:
		0	ADC Results Memory is currently not being initialized, and the ADC is available. If this bit is read as 0 after triggering an auto-initialization of the ADC results memory, then the ADC results memory has been completely initialized to zeros. For devices requiring parity checking on the ADC results memory, the parity bit in the results memory will also be initialized according to the parity polarity. The parity polarity as well as the auto-initialization process is controlled by the System module. Refer to Chapter 2 for more details.
		1	ADC results memory is being initialized, and the ADC is not available for conversion.
15-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	BNDEND		Buffer Boundary End. These bits specify the total number of memory buffers available for storing the ADC conversion results. These bits should be programmed to match the number of ADC conversion result buffers required to be used for the application. Any operation mode read/write:
		0	16 words available for storing ADC conversion results.
		1h	32 words available for storing ADC conversion results.
		2h	64 words available for storing ADC conversion results. This is the maximum configuration since the ADC is 64 words total.
		4h-7h	Reserved. These combinations must not be used.

16.10.22 ADC Event Group Sampling Time Configuration Register (ADEVSAMP)

ADC Event Group Sampling Time Configuration Register (ADEVSAMP) is shown in Figure 16-42 and described in Table 16-28.

Figure 16-42. ADC Event Group Sampling Time Configuration Register (ADEVSAMP) [offset = 60h]

31	12	11	0
Reserved		EV_ACQ	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-28. ADC Event Group Sampling Time Configuration Register (ADEVSAMP) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-0	EV_ACQ		Event Group Acquisition Time. These bits define the sampling window (SW) for the Event Group conversions. SW = EV_ACQ + 2 in terms of ADCLK cycles. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that SW >= 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be guaranteed by configuring the EV_ACQ value properly considering the frequency of the ADCLK signal. Refer to the device datasheet to determine the minimum sampling time for this device.

16.10.23 ADC Group1 Sampling Time Configuration Register (ADG1SAMP)

ADC Group1 Sampling Time Configuration Register (ADG1SAMP) is shown in Figure 16-43 and described in Table 16-29.

Figure 16-43. ADC Group1 Sampling Time Configuration Register (ADG1SAMP) [offset = 64h]

31	12	11	0
Reserved		G1_ACQ	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-29. ADC Group1 Sampling Time Configuration Register (ADG1SAMP) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-0	G1_ACQ		Group1 Acquisition Time. These bits define the sampling window (SW) for the Group1 conversions. SW = G1_ACQ + 2 in terms of ADCLK cycles. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that SW >= 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be guaranteed by configuring the G1_ACQ value properly considering the frequency of the ADCLK signal. Refer to the device datasheet to determine the minimum sampling time for this device.

16.10.24 ADC Group2 Sampling Time Configuration Register (ADG2SAMP)

ADC Group2 Sampling Time Configuration Register (ADG2SAMP) is shown in Figure 16-44 and described in Table 16-30.

Figure 16-44. ADC Group2 Sampling Time Configuration Register (ADG2SAMP) [offset = 68h]

31	12	11	0
Reserved		G2_ACQ	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-30. ADC Group2 Sampling Time Configuration Register (ADG2SAMP) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-0	G2_ACQ		Group2 Acquisition Time. These bits define the sampling window (SW) for the Group2 conversions. SW = G2_ACQ + 2 in terms of ADCLK cycles. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that SW >= 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be guaranteed by configuring the G2_ACQ value properly considering the frequency of the ADCLK signal. Refer to the device datasheet to determine the minimum sampling time for this device.

16.10.25 ADC Event Group Status Register (ADEVSR)

ADC Event Group Status Register (ADEVSR) is shown in Figure 16-45 and described in Table 16-31.

Figure 16-45. ADC Event Group Status Register (ADEVSR) [offset = 6Ch]

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		EV_MEM_EMPTY	EV_BUSY	EV_STOP	EV_END
R-0		R-1	R-0	R-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 16-31. ADC Event Group Status Register (ADEVSR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	EV_MEM_EMPTY		Event Group Results Memory Empty. This bit can be effectively used only when the conversion results are read out of the Event Group results memory in the "read from FIFO" mode.
			Any operation mode read:
		0	The Event Group results memory has valid conversion results.
		1	The Event Group results memory is empty, or does not contain any unread conversion results.
2	EV_BUSY		Event Group Conversion Busy.
			Any operation mode read:
		0	Event Group conversions are neither in progress nor frozen.
		1	Event Group conversions are either in progress, or are frozen for servicing some other group. This bit will always be set when the Event Group is configured to be in the continuous conversion mode.
1	EV_STOP		Event Group Conversion Stopped.
			Any operation mode read:
		0	Event Group conversions are not currently frozen.
		1	Event Group conversions are currently frozen.
0	EV_END		Event Group Conversions Ended.
			Any operation mode read:
		0	Event Group conversions have either not been started or have not yet completed since the last time this status bit was cleared.
		1	The conversion for all the channels selected in the Event Group has completed. This bit can be cleared under the following conditions: • By reading a conversion result from the Event Group results memory in the "read from FIFO" mode. • By writing a new value to the Event Group channel select register ADEVSEL. • By writing a 1 to this bit. • By disabling the ADC module by clearing the ADC_EN bit in the ADC operating mode control register (ADOPMODECR).

16.10.26 ADC Group1 Status Register (ADG1SR)

ADC Group1 Status Register (ADG1SR) is shown in Figure 16-46 and described in Table 16-32.

Figure 16-46. ADC Group1 Status Register (ADG1SR) [offset = 70h]

31	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		G1_MEM_EMPTY	G1_BUSY	G1_STOP	G1_END
R-0		R-1	R-0	R-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 16-32. ADC Group1 Status Register (ADG1SR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	G1_MEM_EMPTY		Group1 Results Memory Empty. This bit can be effectively used only when the conversion results are read out of the Group1 results memory in the "read from FIFO" mode. Any operation mode read:
		0	The Group1 results memory has valid conversion results.
		1	The Group1 results memory is empty, or does not contain any unread conversion results.
2	G1_BUSY		Group1 Conversion Busy. Any operation mode read:
		0	Group1 conversions are neither in progress nor frozen.
		1	Group1 conversions are either in progress, or are frozen for servicing some other group. This bit will always be set when the Group1 is configured to be in the continuous conversion mode.
1	G1_STOP		Group1 Conversion Stopped. Any operation mode read:
		0	Group1 conversions are not currently frozen.
		1	Group1 conversions are currently frozen.
0	G1_END		Group1 Conversions Ended. Any operation mode read:
		0	Group1 conversions have either not been started or have not yet completed since the last time this status bit was cleared.
		1	The conversion for all the channels selected in the Group1 has completed. This bit can be cleared under the following conditions: • By reading a conversion result from the Group1 results memory in the "read from FIFO" mode. • By writing a new value to the Group1 channel select register ADG1SEL. • By writing a 1 to this bit. • By disabling the ADC module by clearing the ADC_EN bit in the ADC operating mode control register (ADOPMODECR).

16.10.27 ADC Group2 Status Register (ADG2SR)

ADC Group2 Status Register (ADG2SR) is shown in Figure 16-47 and described in Table 16-33.

Figure 16-47. ADC Group2 Status Register (ADG2SR) [offset = 74h]

31					8
Reserved					
R-0					
7	4	3	2	1	0
Reserved		G2_MEM_EMPTY	G2_BUSY	G2_STOP	G2_END
R-0		R-1	R-0	R-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 16-33. ADC Group2 Status Register (ADG2SR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3	G2_MEM_EMPTY		Group2 Results Memory Empty. This bit can be effectively used only when the conversion results are read out of the Group2 results memory in the "read from FIFO" mode. Any operation mode read:
		0	The Group2 results memory has valid conversion results.
		1	The Group2 results memory is empty, or does not contain any unread conversion results.
2	G2_BUSY		Group2 Conversion Busy. Any operation mode read:
		0	Group2 conversions are neither in progress nor frozen.
		1	Group2 conversions are either in progress, or are frozen for servicing some other group. This bit will always be set when the Group2 is configured to be in the continuous conversion mode.
1	G2_STOP		Group2 Conversion Stopped. Any operation mode read:
		0	Group2 conversions are not currently frozen.
		1	Group2 conversions are currently frozen.
0	G2_END		Group2 Conversions Ended. Any operation mode read:
		0	Group2 conversions have either not been started or have not yet completed since the last time this status bit was cleared.
		1	The conversion for all the channels selected in the Group2 has completed. This bit can be cleared under the following conditions: • By reading a conversion result from the Group2 results memory in the "read from FIFO" mode. • By writing a new value to the Group2 channel select register ADG2SEL. • By writing a 1 to this bit. • By disabling the ADC module by clearing the ADC_EN bit in the ADC operating mode control register (ADOPMODECR).

16.10.28 ADC Event Group Channel Select Register (ADEVSEL)

ADC Event Group Channel Select Register (ADEVSEL) is shown in Figure 16-48 and described in Table 16-34.

NOTE: Clearing ADEVSEL During a Conversion

Writing 0000h to ADEVSEL stops the Event Group conversions. This does not cause the ADC Event Group results Memory pointer or the Event Group Threshold Register to be reset.

NOTE: Writing A Non-Zero Value To ADEVSEL During a Conversion

Writing a new value to ADEVSEL while a Channel in Event Group is being converted results in a new conversion sequence starting immediately with the highest priority channel in the new ADEVSEL selection. This also causes the ADC Event Group Results Memory pointer to be reset so that the memory allocated for storing the Event Group conversion results gets overwritten. Care should be taken to re-program the corresponding Interrupt Threshold Counter again so that correct number of conversions happen before a Threshold interrupt is generated.

Figure 16-48. ADC Event Group Channel Select Register (ADEVSEL) [offset = 78h]

31	16	15	0
Reserved		EV_SEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-34. ADC Event Group Channel Select Register (ADEVSEL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	EV_SEL		Event Group channels selected.
			Any operation mode read/write:
		0	No ADC input channel is selected for conversion in the Event Group.
		Non-zero	The channels marked by the bit positions that are set to 1 will be converted in ascending order when the Event Group is triggered.

16.10.29 ADC Group1 Channel Select Register (ADG1SEL)

ADC Group1 Channel Select Register (ADG1SEL) is shown in Figure 16-49 and described in Table 16-35.

NOTE: Clearing ADG1SEL During a Conversion

Writing 0000h to ADG1SEL stops the Group1 conversions. This does not cause the ADC Group1 Results Memory pointer or the Group1 Threshold Register to be reset.

NOTE: Writing A Non-Zero Value To ADG1SEL During a Conversion

Writing a new value to ADG1SEL while a Channel in Group1 is being converted results in a new conversion sequence starting immediately with the highest priority channel in the new ADG1SEL selection. This also causes the ADC Group1 Results Memory pointer to be reset so that the memory allocated for storing the Group1 conversion results gets overwritten. Care should be taken to re-program the corresponding Interrupt Threshold Counter again so that the correct number of conversions happen before a Threshold interrupt is generated.

Figure 16-49. ADC Group1 Channel Select Register (ADG1SEL) [offset = 7Ch]

31	16	15	0
Reserved		G1_SEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-35. ADC Group1 Channel Select Register (ADG1SEL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	G1_SEL		Group1 channels selected.
			Any operation mode read/write:
		0	No ADC input channel is selected for conversion in the Group1.
		Non-zero	The channels marked by the bit positions that are set to 1 will be converted in ascending order when the Group1 is triggered.

16.10.30 ADC Group2 Channel Select Register (ADG2SEL)

ADC Group2 Channel Select Register (ADG2SEL) is shown in Figure 16-50 and described in Table 16-36.

NOTE: Clearing ADG2SEL During a Conversion

Writing 0000h to ADG2SEL stops the Group2 conversions. This does not cause the ADC Group2 Results Memory pointer or the Group2 Threshold Register to be reset.

NOTE: Writing A Non-Zero Value To ADG2SEL During a Conversion

Writing a new value to ADG2SEL while a Channel in Group2 is being converted results in a new conversion sequence starting immediately with the highest priority channel in the new ADG2SEL selection. This also causes the ADC Group2 Results Memory pointer to be reset so that the memory allocated for storing the Group2 conversion results gets overwritten. Care should be taken to re-program the corresponding Interrupt Threshold Counter again so that correct number of conversions happen before a Threshold interrupt is generated.

Figure 16-50. ADC Group2 Channel Select Register (ADG2SEL) [offset = 80h]

31	16	15	0
Reserved		G2_SEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-36. ADC Group2 Channel Select Register (ADG2SEL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	G2_SEL		Group2 channels selected.
			Any operation mode read/write:
		0	No ADC input channel is selected for conversion in the Group2.
		Non-zero	The channels marked by the bit positions that are set to 1 will be converted in ascending order when the Group2 is triggered.

16.10.31 ADC Calibration and Error Offset Correction Register (ADCALR)

ADC Calibration and Error Offset Correction Register (ADCALR) is shown in Figure 16-51 and Figure 16-52, and described in Table 16-37. As shown, the format of the ADCALR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 16-51. 12-bit ADC Calibration and Error Offset Correction Register (ADCALR) [offset = 84h]

31	12	11	0
Reserved		ADCALR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 16-52. 10-bit ADC Calibration and Error Offset Correction Register (ADCALR) [offset = 84h]

31	10	9	0
Reserved		ADCALR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-37. ADC Calibration and Error Offset Correction Register (ADCALR) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
ADCALR		ADC Calibration Result and Offset Error Correction Value. The actual size of the ADCALR field is 12 bits or 10 bits depending on whether the ADC is configured to be in 12-bit or 10-bit resolution mode, respectively. Bits 11-10 are reserved when the module is configured as a 10-bit ADC module. The ADC module writes the results of the calibration conversions to this register. The application is required to use these conversion results and determine the ADC offset error. The application can then compute the correction for the offset error and this correction value needs to be written back to the ADCALR register in the 2's complement form. During normal conversion (when calibration is disabled), the ADCALR register contents are automatically added to each digital output from the ADC core before it is stored in the ADC results memory. For more details on error calibration, refer to Section 16.7.1.

16.10.32 ADC State Machine Status Register (ADSMSTATE)

Figure 16-53 and Table 16-38 describe the ADSMSTATE register.

Figure 16-53. ADC State Machine Status Register (ADSMSTATE) [offset = 88h]

31	4	3	0
Reserved		SMSTATE	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 16-38. ADC State Machine Status Register (ADSMSTATE) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	SMSTATE		ADC State Machine Current State. These bits reflect the current state of the state machine and are reserved for use by TI for debug purposes.

16.10.33 ADC Channel Last Conversion Value Register (ADLASTCONV)

ADC Channel Last Conversion Value Register (ADLASTCONV) is shown in Figure 16-54 and described in Table 16-39.

Figure 16-54. ADC Channel Last Conversion Value Register (ADLASTCONV) [offset = 8Ch]

31	24	23	0
Reserved		LAST_CONV	
R-0		R-U	

LEGEND: R = Read only; -U = value after reset is unknown; -*n* = value after reset

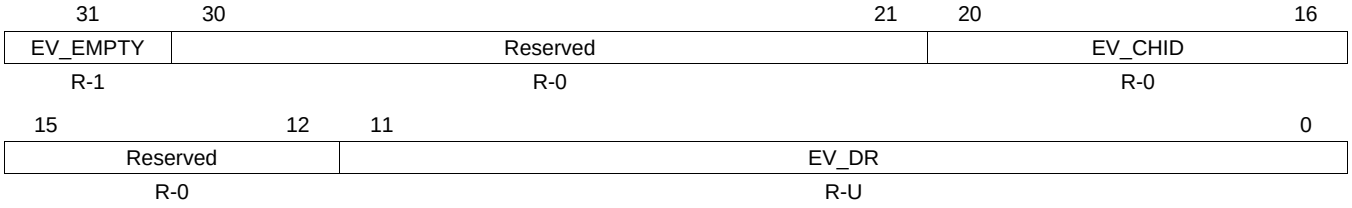
Table 16-39. ADC Channel Last Conversion Value Register (ADLASTCONV) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	LAST_CONV		ADC Input Channel's Last Converted Value. This register indicates whether the last converted value for a particular input channel was lower or higher than the mid-point of the reference voltage. In other words, this register acts as a digital input register and can be read by the application to determine the digital level at the input pins. This data is only valid for an input channel if it has been converted at least once. Any operation mode read for each bit of this register:
		0	A level lower than the midpoint reference voltage was measured at the last conversion for this channel.
		1	A level higher than or equal to the midpoint reference voltage was measured at the last conversion for this channel.

16.10.34 ADC Event Group Results' FIFO Register (ADEVBUFFER)

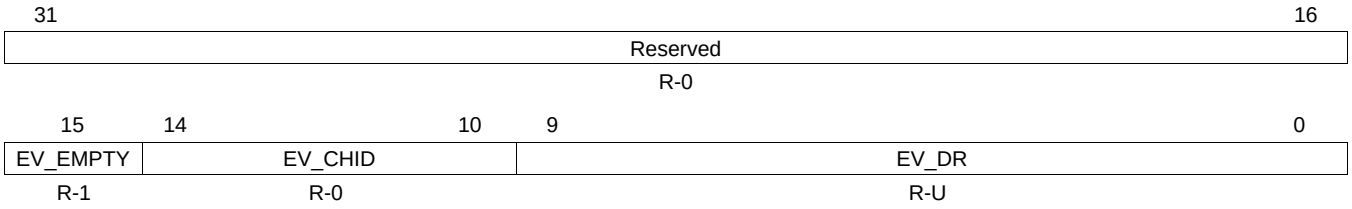
ADC Event Group Results' FIFO Register (ADEVBUFFER) is shown in Figure 16-55 and Figure 16-56, and described in Table 16-40. The format of the data read from the ADEVBUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 16-55. 12-bit ADC Event Group Results' FIFO Register (ADEVBUFFER) [offset = 90h-AFh]



LEGEND: R = Read only; -U = value after reset is unknown; -*n* = value after reset

Figure 16-56. 10-bit ADC Event Group Results' FIFO Register (ADEVBUFFER) [offset = 90h-AFh]



LEGEND: R = Read only; -U = value after reset is unknown; -*n* = value after reset

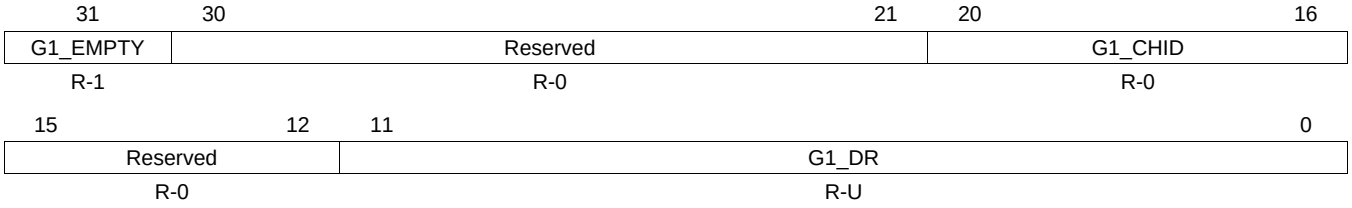
Table 16-40. ADC Event Group Results' FIFO Register (ADEVBUFFER) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
EV_EMPTY		Event Group FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results.
		Any operation mode read:
	0	The data in the EV_DR field of this buffer is valid.
	1	The data in the EV_DR field of this buffer is not valid and there are no valid data in the Event Group results memory.
EV_CHID		Event Group Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results.
		Any operation mode read:
	0	The conversion result in the EV_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Event Group mode control register (ADEVMODECR).
	1h-1Fh	The conversion result in the EV_DR field of this buffer is from the ADC input channel number denoted by the EV_CHID field.
EV_DR		Event Group Digital Conversion Result.
		The Event Group results' FIFO location is aliased eight times, so that any word-aligned read from the address range 0x90 to 0xAF results in one conversion result to be read from the Event Group results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Event Group results' memory with just one instruction.

16.10.35 ADC Group1 Results FIFO Register (ADG1BUFFER)

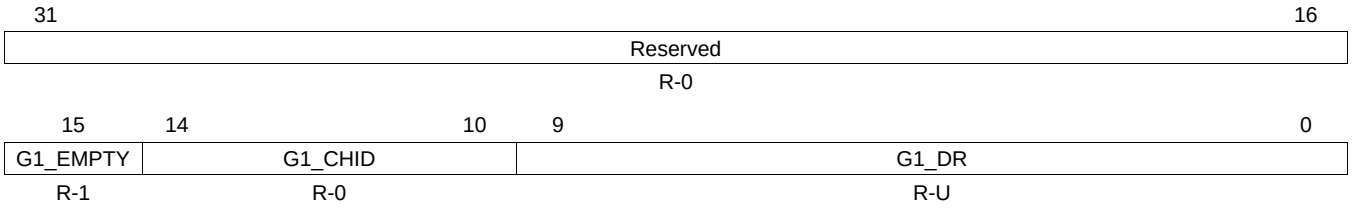
ADC Group1 Results FIFO Register (ADG1BUFFER) is shown in Figure 16-57 and Figure 16-58, and described in Table 16-41. The format of the data read from the ADG1BUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 16-57. 12-bit ADC Group1 Results FIFO Register (ADG1BUFFER) [offset = B0h-CFh]



LEGEND: R = Read only; -U = value after reset is unknown; -*n* = value after reset

Figure 16-58. 10-bit ADC Group1 Results' FIFO Register (ADG1BUFFER) [offset = B0h-CFh]



LEGEND: R = Read only; -U = value after reset is unknown; -*n* = value after reset

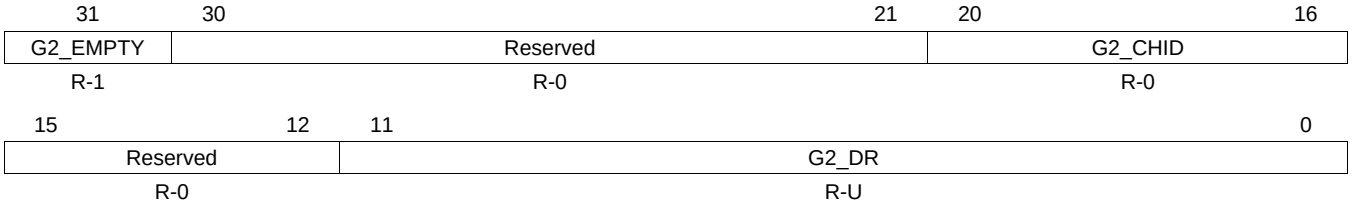
Table 16-41. ADC Group1 Results FIFO Register (ADG1BUFFER) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
G1_EMPTY		Group1 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results.
		Any operation mode read:
	0	The data in the G1_DR field of this buffer is valid.
	1	The data in the G1_DR field of this buffer is not valid and there are no valid data in the Group1 results memory.
G1_CHID		Group1 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results.
		Any operation mode read:
	0	The conversion result in the G1_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Group1 mode control register (ADG1MODECR).
	1h-1Fh	The conversion result in the G1_DR field of this buffer is from the ADC input channel number denoted by the G1_CHID field.
G1_DR		Group1 Digital Conversion Result.
		The Group1 results' FIFO location is aliased eight times, so that any word-aligned read from the address range 0xB0 to 0xCF results in one conversion result to be read from the Group1 results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Group1 results' memory with just one instruction.

16.10.36 ADC Group2 Results FIFO Register (ADG2BUFFER)

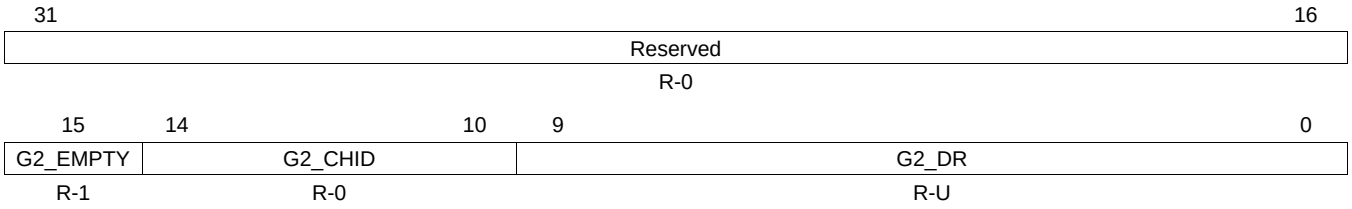
ADC Group2 Results FIFO Register (ADG2BUFFER) is shown in Figure 16-59 and Figure 16-60, and described in Table 16-42. The format of the data read from the ADG2BUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 16-59. 12-bit ADC Group2 Results FIFO Register (ADG2BUFFER) [offset = D0h-EFh]



LEGEND: R = Read only; -U = value after reset is unknown; -*n* = value after reset

Figure 16-60. 10-bit ADC Group2 Results' FIFO Register (ADG2BUFFER) [offset = D0h-EFh]



LEGEND: R = Read only; -U = value after reset is unknown; -*n* = value after reset

Table 16-42. ADC Group2 Results FIFO Register (ADG2BUFFER) Field Descriptions

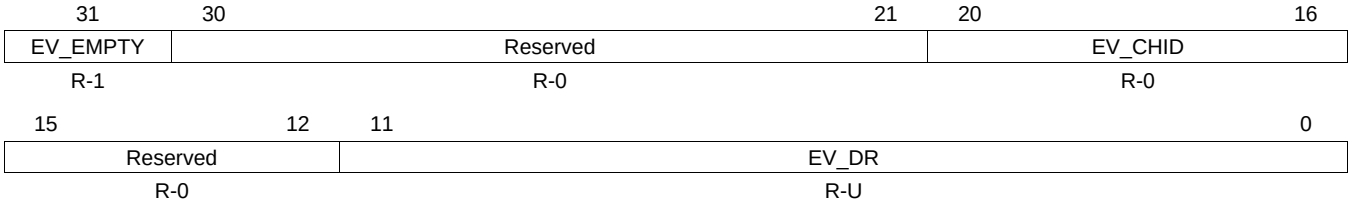
Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
G2_EMPTY		Group2 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results.
		Any operation mode read:
	0	The data in the G2_DR field of this buffer is valid.
	1	The data in the G2_DR field of this buffer is not valid and there are no valid data in the Group2 results memory.
G2_CHID		Group2 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results.
		Any operation mode read:
	0	The conversion result in the G2_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Group2 mode control register (ADG2MODECR).
	1h-1Fh	The conversion result in the G2_DR field of this buffer is from the ADC input channel number denoted by the G2_CHID field.
G2_DR		Group2 Digital Conversion Result.
		The Group2 results' FIFO location is aliased eight times, so that any word-aligned read from the address range 0xD0 to 0xEF results in one conversion result to be read from the Group2 results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Group2 results' memory with just one instruction.

16.10.37 ADC Event Group Results Emulation FIFO Register (ADEVEMUBUFFER)

ADC Event Group Results Emulation FIFO Register (ADEVEMUBUFFER) is shown in Figure 16-61 and Figure 16-62, and described in Table 16-43. As shown, the format of the data read from the ADEVEMUBUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

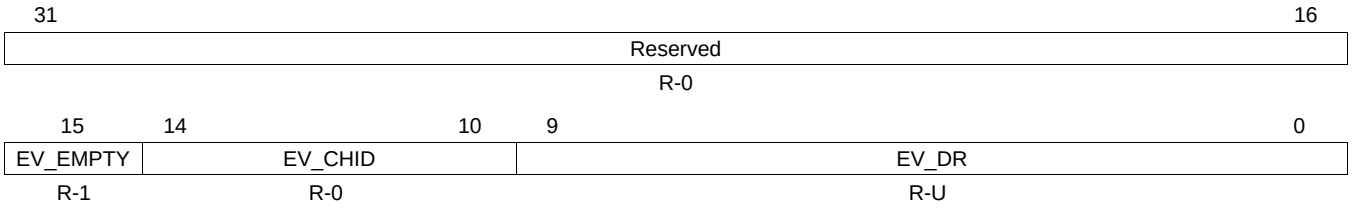
A read from this location also gives out one conversion result from the Event Group results' memory along with the EV_EMPTY status bit and the optional channel id. However, this read will not affect any of the status flags in the Event Group interrupt flag register or the Event Group status register. This register is useful for debuggers.

Figure 16-61. 12-bit ADC Event Group Results Emulation FIFO Register (ADEVEMUBUFFER) [offset = F0h]



LEGEND: R = Read only; -U = value after reset is unknown; *-n* = value after reset

Figure 16-62. 10-bit ADC Event Group Results Emulation FIFO Register (ADEVEMUBUFFER) [offset = F0h]



LEGEND: R = Read only; -U = value after reset is unknown; *-n* = value after reset

Table 16-43. ADC Event Group Results Emulation FIFO Register (ADEVEMUBUFFER) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
EV_EMPTY		Event Group FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results.
		Any operation mode read:
	0	The data in the EV_DR field of this buffer is valid.
	1	The data in the EV_DR field of this buffer is not valid and there are no valid data in the Event Group results memory.
EV_CHID		Event Group Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results.
		Any operation mode read:
	0	The conversion result in the EV_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Event Group operating mode control register (ADEVMODECR).
	1h-1Fh	The conversion result in the EV_DR field of this buffer is from the ADC input channel number denoted by the EV_CHID field.
EV_DR		Event Group Digital Conversion Result.
		These bits contain the digital result output from the Event Group FIFO buffer. The result can be presented in an 8-bit, 10-bit, or 12-bit format for a 12-bit ADC module, or in an 8-bit or 10-bit format for a 10-bit ADC module. The conversion result data is automatically shifted right by the appropriate number of bits when using a reduced-size data format with the upper bits reading as zeros.

16.10.38 ADC Group1 Results Emulation FIFO Register (ADG1EMUBUFFER)

ADC Group1 Results Emulation FIFO Register (ADG1EMUBUFFER) is shown in Figure 16-63 and Figure 16-64, described in Table 16-44. As shown, the format of the data read from the ADG1EMUBUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

A read from this location also gives out one conversion result from the Group1 results' memory along with the G1_EMPTY status bit and the optional channel id. However, this read will not affect any of the status flags in the Group1 interrupt flag register or the Group1 status register. This register is useful for debuggers.

Figure 16-63. 12-bit ADC Group1 Results Emulation FIFO Register (ADG1EMUBUFFER) [offset = F4h]

31	30	21	20	16
G1_EMPTY	Reserved		G1_CHID	
R-1	R-0		R-0	

15	12	11	0
Reserved		G1_DR	
R-0		R-U	

LEGEND: R = Read only; -U = value after reset is unknown; *-n* = value after reset

Figure 16-64. 10-bit ADC Group1 Results Emulation FIFO Register (ADG1EMUBUFFER) [offset = F4h]

31	16
Reserved	
R-0	

15	14	10	9	0
G1_EMPTY	G1_CHID		G1_DR	
R-1	R-0		R-U	

LEGEND: R = Read only; -U = value after reset is unknown; *-n* = value after reset

Table 16-44. ADC Group1 Results Emulation FIFO Register (ADG1EMUBUFFER) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
G1_EMPTY		Group1 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results.
		Any operation mode read:
	0	The data in the G1_DR field of this buffer is valid.
	1	The data in the G1_DR field of this buffer is not valid and there are no valid data in the Group1 results memory.
G1_CHID		Group1 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results.
		Any operation mode read:
	0	The conversion result in the G1_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Group1 operating mode control register (ADG1MODECR).
	1h-1Fh	The conversion result in the G1_DR field of this buffer is from the ADC input channel number denoted by the G1_CHID field.
G1_DR		Group1 Digital Conversion Result.
		These bits contain the digital result output from the Group 1 FIFO buffer. The result can be presented in an 8-bit, 10-bit, or 12-bit format for a 12-bit ADC module, or in an 8-bit or 10-bit format for a 10-bit ADC module. The conversion result data is automatically shifted right by the appropriate number of bits when using a reduced-size data format with the upper bits reading as zeros.

16.10.39 ADC Group2 Results Emulation FIFO Register (ADG2EMUBUFFER)

ADC Group2 Results Emulation FIFO Register (ADG2EMUBUFFER) is shown in Figure 16-65 and Figure 16-66, described in Table 16-45. As shown, the format of the data read from the ADG2EMUBUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

A read from this location also gives out one conversion result from the Group2 results' memory along with the G2_EMPTY status bit and the optional channel id. However, this read will not affect any of the status flags in the Group2 interrupt flag register or the Group2 status register. This register is useful for debuggers.

Figure 16-65. 12-bit ADC Group2 Results Emulation FIFO Register (ADG2EMUBUFFER) [offset = F8h]

31	30	21	20	16
G2_EMPTY	Reserved		G2_CHID	
R-1	R-0		R-0	

15	12	11	0
Reserved		G2_DR	
R-0		R-U	

LEGEND: R = Read only; -U = value after reset is unknown; *-n* = value after reset

Figure 16-66. 10-bit ADC Group2 Results Emulation FIFO Register (ADG2EMUBUFFER) [offset = F8h]

31	16
Reserved	
R-0	

15	14	10	9	0
G2_EMPTY	G2_CHID		G2_DR	
R-1	R-0		R-U	

LEGEND: R = Read only; -U = value after reset is unknown; *-n* = value after reset

Table 16-45. ADC Group2 Results Emulation FIFO Register (ADG2EMUBUFFER) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
G2_EMPTY		Group2 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results.
		Any operation mode read:
	0	The data in the G2_DR field of this buffer is valid.
	1	The data in the G2_DR field of this buffer is not valid and there are no valid data in the Group2 results memory.
G2_CHID		Group2 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results.
		Any operation mode read:
	0	The conversion result in the G2_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Group2 operating mode control register (ADG2MODECR).
	1h-1Fh	The conversion result in the G2_DR field of this buffer is from the ADC input channel number denoted by the G2_CHID field.
G2_DR		Group2 Digital Conversion Result.
		These bits contain the digital result output from the Group 2 FIFO buffer. The result can be presented in an 8-bit, 10-bit, or 12-bit format for a 12-bit ADC module, or in an 8-bit or 10-bit format for a 10-bit ADC module. The conversion result data is automatically shifted right by the appropriate number of bits when using a reduced-size data format with the upper bits reading as zeros.

16.10.40 ADC ADEVT Pin Direction Control Register (ADEVTDIR)

ADC ADEVT Pin Direction Control Register (ADEVTDIR) is shown in Figure 16-67 and described in Table 16-46.

Figure 16-67. ADC ADEVT Pin Direction Control Register (ADEVTDIR) [offset = FCh]

31	1	0
Reserved		ADEVT_DIR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-46. ADC ADEVT Pin Direction Control Register (ADEVTDIR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADEVT_DIR		ADEVT Pin Direction.
			Any operating mode read/write:
		0	ADEVT is an input pin; the output buffer is disabled.
		1	ADEVT is an output pin; the output buffer is enabled.

16.10.41 ADC ADEVT Pin Output Value Control Register (ADEVTOUT)

ADC ADEVT Pin Output Value Control Register (ADEVTOUT) is shown in Figure 16-68 and described in Table 16-47.

Figure 16-68. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) [offset = 100h]

31	1	0
Reserved		ADEVT_OUT
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-47. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADEVT_OUT		ADEVT Pin Output Value. This bit determines the logic level to be output to the ADEVT pin when the pin is configured to be an output pin.
			Any operating mode read/write:
		0	Output logic LOW on the ADEVT pin.
		1	Output logic HIGH on the ADEVT pin.

16.10.42 ADC ADEVT Pin Input Value Register (ADEVTIN)

ADC ADEVT Pin Input Value Register (ADEVTIN) is shown in Figure 16-69 and described in Table 16-48.

Figure 16-69. ADC ADEVT Pin Input Value Register (ADEVTIN) [offset = 104h]

31	1	0
Reserved		ADEVT_IN
R-0		R-U

LEGEND: R = Read only; -U = value after reset is unknown; -*n* = value after reset

Table 16-48. ADC ADEVT Pin Input Value Register (ADEVTIN) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADEVT_IN		ADEVT Pin Input Value. This is a read-only bit which reflects the logic level on the ADEVT pin.
			Any operating mode read:
		0	Logic LOW present on the ADEVT pin.
		1	Logic HIGH present on the ADEVT pin.

16.10.43 ADC ADEVT Pin Set Register (ADEVTSET)

ADC ADEVT Pin Set Register (ADEVTSET) is shown in Figure 16-70 and described in Table 16-49.

Figure 16-70. ADC ADEVT Pin Set Register (ADEVTSET) [offset = 108h]

31	1	0
Reserved		ADEVT_SET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-49. ADC ADEVT Pin Set Register (ADEVTSET) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADEVT_SET		ADEVT Pin Set. This bit drives the output of the ADEVT pin high. A read from this bit always returns the current state of the ADEVT pin.
			Any operating mode read/write:
		0	Output value on the ADEVT pin is unchanged.
		1	Output logic HIGH on the ADEVT pin, if the pin is configured to be an output pin.

16.10.44 ADC ADEVT Pin Clear Register (ADEVTCLR)

ADC ADEVT Pin Clear Register (ADEVTCLR) is shown in Figure 16-71 and described in Table 16-50.

Figure 16-71. ADC ADEVT Pin Clear Register (ADEVTCLR) [offset = 10Ch]

31	1	0
Reserved		ADEVT_CLR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-50. ADC ADEVT Pin Clear Register (ADEVTCLR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADEVT_CLR		ADEVT Pin Clear. A read from this bit always returns the current state of the ADEVT pin.
			Any operating mode read/write:
		0	Output value on the ADEVT pin is unchanged.
		1	Output logic LOW on the ADEVT pin, if the pin is configured to be an output pin.

16.10.45 ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR)

ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) is shown in Figure 16-72 and described in Table 16-51.

Figure 16-72. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) [offset = 110h]

31	1	0
Reserved		ADEVT_PDR
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-51. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADEVT_PDR		ADEVT Pin Open Drain Enable. This bit enables the open-drain capability for the ADEVT pin if it is configured to be an output and a logic HIGH is being driven on to the pin.
			Any operating mode read/write:
		0	Output value on the ADEVT pin is logic HIGH.
		1	The ADEVT pin is tristated.

16.10.46 ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS)

ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) is shown in Figure 16-73 and described in Table 16-52.

Figure 16-73. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) [offset = 114h]

31	1	0
Reserved		ADEVT_PDIS
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-52. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADEVT_PDIS		ADEVT Pin Pull Control Disable. This bit enables or disables the pull control on the ADEVT pin if it is configured to be an input pin.
			Any operating mode read/write:
		0	Pull on ADEVT pin is enabled.
		1	Pull on ADEVT pin is disabled.

16.10.47 ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL)

ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) is shown in Figure 16-74 and described in Table 16-53.

Figure 16-74. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) [offset = 118h]

31	1	0
Reserved		ADEVT_PSEL
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-53. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ADEVT_PSEL		ADEVT Pin Pull Control Select. This bit selects a pull-down or pull-up on the ADEVT pin if it is configured to be an input pin.
			Any operating mode read/write:
		0	Pull down is selected on ADEVT pin.
		1	Pull up is selected on ADEVT pin.

16.10.48 ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN)

ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) is shown in Figure 16-75 and described in Table 16-54.

Figure 16-75. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) [offset = 11Ch]

31	16
Reserved	
R-0	

15	8	7	1	0
EV_SAMP_DIS_CYC		Reserved		EV_SAMP_DIS_EN
R/W-0		R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-54. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	EV_SAMP_DIS_CYC		Event Group sample cap discharge cycles. These bits specify the duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	EV_SAMP_DIS_EN		Event Group sample cap discharge enable.
			Any operation mode read/write:
		0	Event Group sample cap discharge mode is disabled.
		1	Event Group sample cap discharge mode is enabled. The ADC internal sampling capacitor is connected to the V_{REFLO} reference voltage for a duration specified by the EV_SAMP_DIS_CYC field. After this discharge time has expired the selected ADC input channel is sampled and converted normally based on the Event Group settings.

16.10.49 ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN)

ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) is shown in Figure 16-76 and described in Table 16-55.

Figure 16-76. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) [offset = 120h]

31	16
Reserved	
R-0	

15	8	7	1	0
G1_SAMP_DIS_CYC		Reserved		G1_SAMP_DIS_EN
R/W-0		R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-55. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	G1_SAMP_DIS_CYC		Group1 sample cap discharge cycles. These bits specify the duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	G1_SAMP_DIS_EN		Group1 sample cap discharge enable. Any operation mode read/write:
		0	Group1 sample cap discharge mode is disabled.
		1	Group1 sample cap discharge mode is enabled. The ADC internal sampling capacitor is connected to the V_{REFLO} reference voltage for a duration specified by the G1_SAMP_DIS_CYC field. After this discharge time has expired the selected ADC input channel is sampled and converted normally based on the Group1 settings.

16.10.50 ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN)

ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) is shown in Figure 16-77 and described in Table 16-56.

Figure 16-77. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) [offset = 124h]

31	16
Reserved	
R-0	

15	8	7	1	0
G2_SAMP_DIS_CYC		Reserved		G2_SAMP_DIS_EN
R/W-0		R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-56. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	G2_SAMP DIS CYC		Group2 sample cap discharge cycles. These bits specify the duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	G2_SAMP_DIS_EN		Group2 sample cap discharge enable. Any operation mode read/write:
		0	Group2 sample cap discharge mode is disabled.
		1	Group2 sample cap discharge mode is enabled. The ADC internal sampling capacitor is connected to the V_{REFLO} reference voltage for a duration specified by the G2_SAMP_DIS_CYC field. After this discharge time has expired the selected ADC input channel is sampled and converted normally based on the Group2 settings.

16.10.51 ADC Magnitude Compare Interrupt x Control Registers (ADMAGINTxCR)

ADC Magnitude Compare Interrupt x Control Registers (ADMAGINTxCR) are shown in Figure 16-78 and Figure 16-79, and described in Table 16-57. As shown, the format of the ADMAGINTxCR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module. The ADC module supports up to three magnitude compare interrupts. These registers are at offset addresses 128h, 130h, and 138h.

Figure 16-78. 12-bit ADC Magnitude Compare Interrupt x Control Registers (ADMAGINTxCR) [offset = 128h-138h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				MAG_THRx											
R-0				R/W-0											

15	14	13	12	11	10	9	8
CHN_THR_COMPx	CMP_GE_LTx	Reserved	COMP_CHIDx				
R/W-0	R/W-0	R-0	R/W-0				

7	6	5	4	3	2	1	0
Reserved			MAG_CHIDx				
R-0			R/W-0				

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 16-79. 10-bit ADC Magnitude Compare Interrupt x Control Registers (ADMAGINTxCR) [offset = 128h-138h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Rsvd	MAG_CHIDx					MAG_THRx									
R-0	R/W-0					R/W-0									

15	14	13	12	11	10	9	8
Reserved			COMP_CHIDx				
R-0			R/W-0				

7	6	5	4	3	2	1	0
Reserved						CHN_THR_COMPx	CMP_GE_LTx
R-0						R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-57. ADC Magnitude Compare Interrupt x Control Registers (ADMAGINTxCR) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
MAG_CHIDx		These bits specify the channel number from 0 to 31 for which the conversion result needs to be monitored by the ADC.
MAG_THRx		These bits specify the 12-bit or 10-bit compare value that the ADC will use for the comparison with the MAG_CHIDx channel's conversion result.
COMP_CHIDx		These bits specify the channel number from 0 to 31 whose last conversion result is compared with the MAG_CHIDx channel's conversion result.
CHN_THR_COMPx		Channel OR Threshold comparison. Any operation mode read/write:
	0	The ADC module will compare the MAG_CHIDx channel's conversion result with the fixed threshold value specified by the MAG_THRx field
	1	The ADC module will compare the MAG_CHIDx channel's conversion result with the last conversion result for the COMP_CHIDx channel. Both the MAG_CHIDx and the COMP_CHIDx channel must have been converted at least once for the ADC to perform the comparison.
CMP_GE_LTx		"Greater than or equal to" OR "Less than" comparison operator. Any operation mode read/write:
	0	The ADC module will check if the conversion result is lower than the reference value (fixed threshold or COMP_CHIDx conversion result).
	1	The ADC module will check if the conversion result is greater than or equal to the reference value (fixed threshold or COMP_CHIDx conversion result).

16.10.52 ADC Magnitude Compare Interrupt x Mask Register (ADMAGxMASK)

ADC Magnitude Compare Interrupt x Mask Register (ADMAGxMASK) is shown in Figure 16-80and Figure 16-81, and described in Table 16-58. As shown, the format of the ADMAGxMASK is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module. There are three mask registers for the three magnitude compare interrupts. These registers are at offset addresses 12Ch, 134h, and 13Ch.

Figure 16-80. 12-bit ADC Magnitude Compare Interrupt x Mask Register (ADMAGxMASK) [offset = 12Ch-13Ch]

31	12	11	0
Reserved		MAG_INTx_MASK	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 16-81. 10-bit ADC Magnitude Compare Interrupt x Mask Register (ADMAGxMASK) [offset = 12Ch-13Ch]

31	10	9	0
Reserved		MAG_INTx_MASK	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-58. ADC Magnitude Compare Interrupt x Mask Register (ADMAGxMASK) Field Descriptions

Field	Value	Description
Reserved	0	Reads return 0. Writes have no effect.
MAG_INTx_MASK		These bits specify the mask for the comparison in order to generate the magnitude compare interrupt # x.
		Any operation mode read/write:
	0	The ADC module will not mask the corresponding bit for the comparison.
	1	The ADC module will mask the corresponding bit for the comparison.

16.10.53 ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET)

ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET) is shown in Figure 16-82 and described in Table 16-59.

Figure 16-82. ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET) [offset = 158h]

31	3	2	0
Reserved		MAG_INT_ENA_SET	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-59. ADC Magnitude Compare Interrupt Enable Set Register (ADMAGINTENASET) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	MAG_INT_ENA_SET		Each of these three bits, when set, enable the corresponding magnitude compare interrupt.
			Any operation mode read/write for each bit:
		0	The enable status of the corresponding magnitude compare interrupt is left unchanged.
		1	The corresponding magnitude compare interrupt is enabled.

16.10.54 ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR)

ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR) is shown in Figure 16-83 and described in Table 16-60.

Figure 16-83. ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR) [offset = 15Ch]

31	3	2	0
Reserved		MAG_INT_ENA_CLR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-60. ADC Magnitude Compare Interrupt Enable Clear Register (ADMAGINTENACLR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	MAG_INT_ENA_CLR		Each of these three bits, when set, enable the corresponding magnitude compare interrupt.
			Any operation mode read/write for each bit:
		0	The enable status of the corresponding magnitude compare interrupt is left unchanged.
		1	The corresponding magnitude compare interrupt is disabled.

16.10.55 ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG)

ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG) is shown in Figure 16-84 and described in Table 16-61.

Figure 16-84. ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG) [offset = 160h]

31	3	2	0
Reserved		MAG_INT_FLG	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-61. ADC Magnitude Compare Interrupt Flag Register (ADMAGINTFLG) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	MAG_INT_FLG		Magnitude Compare Interrupt Flags. These bits can be polled by the application to determine if the magnitude compares have been evaluated as true. When a magnitude compare interrupt flag is set, the corresponding magnitude compare interrupt will be generated if enabled. Any operation mode, for each bit:
		0	Read: The condition for the corresponding magnitude threshold interrupt was false. Write: The corresponding flag is left unchanged.
		1	Read: The condition for the corresponding magnitude threshold interrupt was true. Write: The corresponding flag is cleared. The flag can also be cleared by reading from the magnitude compare interrupt offset register.

16.10.56 ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF)

ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF) is shown in Figure 16-85 and described in Table 16-62.

Figure 16-85. ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF) [offset = 164h]

31	4	3	0
Reserved		MAG_INT_OFF	
R-0		R/C-0	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

Table 16-62. ADC Magnitude Compare Interrupt Offset Register (ADMAGINTOFF) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MAG_INT_OFF		Magnitude Compare Interrupt Offset. This field indexes the currently highest-priority magnitude compare interrupt. Interrupt 1 has the highest priority and interrupt 3 has the lowest priority among the magnitude compare interrupts. Writes to these bits have no effect. A read from this register clears this register as well as the corresponding magnitude compare interrupt flag in the ADMAGINTFLG register. However, a read from this register in emulation mode does not affect this register or the interrupt status flags. Any operation mode read:
		0	No magnitude compare interrupt is pending.
		1h	Magnitude compare interrupt # 1 is pending.
		2h	Magnitude compare interrupt # 2 is pending.
		3h	Magnitude compare interrupt # 3 is pending.
		4h-Fh	Reserved. These combinations do not occur.

16.10.57 ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR)

ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) is shown in Figure 16-86 and described in Table 16-63.

Figure 16-86. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) [offset = 168h]

31	1	0
Reserved		EV_FIFO_RESET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-63. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	EV_FIFO_RESET		ADC Event Group FIFO Reset. The application can set this bit in case of an overrun condition. This allows the ADC module to overwrite the contents of the Event Group results memory starting from the first location. When this bit is set to 1, the ADC module resets its internal Event Group results memory pointers. Then this bit automatically gets cleared, so that the ADC module allows the Event Group results memory to be overwritten only once each time this bit is set to 1. As a result, the EV_FIFO_RESET bit will always be read as a 0. The EV_FIFO_RESET bit will only have the desired effect when the Event Group results memory is in an overrun condition. It must be used when the data already available in the results memory can be discarded. If the application needs the Event Group memory to always be overwritten with the latest available conversion results, then the OVR_EV_RAM_IGN bit in the Event Group operating mode control register (ADEVMODECR) needs to be set to 1.

16.10.58 ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR)

ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) is shown in Figure 16-87 and described in Table 16-64.

Figure 16-87. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) [offset = 16Ch]

31	1	0
Reserved		G1_FIFO_RESET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-64. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	G1_FIFO_RESET	0-1	ADC Group1 FIFO Reset. The application can set this bit in case of an overrun condition. This allows the ADC module to overwrite the contents of the Group1 results memory starting from the first location. When this bit is set to 1, the ADC module resets its internal Group1 results memory pointers. Then this bit automatically gets cleared, so that the ADC module allows the Group1 results memory to be overwritten only once each time this bit is set to 1. As a result, the G1_FIFO_RESET bit will always be read as a 0. The G1_FIFO_RESET bit will only have the desired effect when the Group1 results memory is in an overrun condition. It must be used when the data already available in the results memory can be discarded. If the application needs the Group1 memory to always be overwritten with the latest available conversion results, then the OVR_G1_RAM_IGN bit in the Group1 operating mode control register (ADG1MODECR) needs to be set to 1.

16.10.59 ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR)

ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) is shown in Figure 16-88 and described in Table 16-65.

Figure 16-88. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) [offset = 170h]

31	1	0
Reserved		G2_FIFO_RESET
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-65. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	G2_FIFO_RESET	0-1	ADC Group2 FIFO Reset. The application can set this bit in case of an overrun condition. This allows the ADC module to overwrite the contents of the Group2 results memory starting from the first location. When this bit is set to 1, the ADC module resets its internal Group2 results memory pointers. Then this bit automatically gets cleared, so that the ADC module allows the Group2 results memory to be overwritten only once each time this bit is set to 1. As a result, the G2_FIFO_RESET bit will always be read as a 0. The G2_FIFO_RESET bit will only have the desired effect when the Group2 results memory is in an overrun condition. It must be used when the data already available in the results memory can be discarded. If the application needs the Group2 memory to always be overwritten with the latest available conversion results, then the OVR_G2_RAM_IGN bit in the Group2 operating mode control register (ADG2MODECR) needs to be set to 1.

16.10.60 ADC Event Group RAM Write Address Register (ADEVRAMWRADDR)

ADC Event Group RAM Write Address Register (ADEVRAMWRADDR) is shown in Figure 16-89 and described in Table 16-66.

Figure 16-89. ADC Event Group RAM Write Address Register (ADEVRAMWRADDR) [offset = 174h]

31	9	8	0
Reserved		EV_RAM_ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-66. ADC Event Group RAM Write Address Register (ADEVRAMWRADDR) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	EV_RAM_ADDR		Event Group results memory write pointer. This field shows the address of the location where the next Event Group conversion result will be stored. This is specified in terms of the buffer number. The application can read this register to determine the number of valid Event Group conversion results available until that time.

16.10.61 ADC Group1 RAM Write Address Register (ADG1RAMWRADDR)

ADC Group1 RAM Write Address Register (ADG1RAMWRADDR) is shown in Figure 16-90 and described in Table 16-67.

Figure 16-90. ADC Group1 RAM Write Address Register (ADG1RAMWRADDR) [offset = 178h]

31	9	8	0
Reserved		G1_RAM_ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-67. ADC Group1 RAM Write Address Register (ADG1RAMWRADDR) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	G1_RAM_ADDR		Group1 results memory write pointer. This field shows the address of the location where the next Group1 conversion result will be stored. This is specified in terms of the buffer number. The application can read this register to determine the number of valid Group1 conversion results available until that time.

16.10.62 ADC Group2 RAM Write Address Register (ADG2RAMWRADDR)

ADC Group2 RAM Write Address Register (ADG2RAMWRADDR) is shown in Figure 16-91 and described in Table 16-68.

Figure 16-91. ADC Group2 RAM Write Address Register (ADG2RAMWRADDR) [offset = 17Ch]

31	9	8	0
Reserved		G2_RAM_ADDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-68. ADC Group2 RAM Write Address Register (ADG2RAMWRADDR) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	G2_RAM_ADDR		Group2 results memory write pointer. This field shows the address of the location where the next Group2 conversion result will be stored. This is specified in terms of the buffer number. The application can read this register to determine the number of valid Group2 conversion results available until that time.

16.10.63 ADC Parity Control Register (ADPARCR)

ADC Parity Control Register (ADPARCR) is shown in Figure 16-92 and described in Table 16-69.

Figure 16-92. ADC Parity Control Register (ADPARCR) [offset = 180h]

31	16
Reserved	
R-0	

15	9	8	7	4	3	0
Reserved		TEST	Reserved		PARITY_ENA	
R-0		R/WP-0	R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 16-69. ADC Parity Control Register (ADPARCR) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	TEST		This bit maps the parity bits into the ADC results' RAM frame so that the application can access them.
			Any operation mode read, privileged mode write:
		0	The parity bits are not memory-mapped.
		1	The parity bits are memory-mapped.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PARITY_ENA		Enable parity checking. These bits enable the parity check on read operations and the parity calculation on write operations to the ADC results memory.
			If parity checking is enabled and a parity error is detected, the ADC module sends a parity error signal to the System module.
			Any operation mode read, privileged mode write:
		5h	Parity check is disabled.
		All other values	Parity check is enabled.

16.10.64 ADC Parity Error Address Register (ADPARADDR)

ADC Parity Error Address Register (ADPARADDR) is shown inFigure 16-93 and described in Table 16-70.

Figure 16-93. ADC Parity Error Address Register (ADPARADDR) [offset = 184h]

31	16
Reserved	
R-0	

15	12	11	2	1	0
Reserved		ERROR_ADDRESS		Reserved	
R-0		R-U		R-0	

LEGEND: R/W = Read/Write; R = Read only; -U = value after reset is unknown; -*n* = value after reset

Table 16-70. ADC Parity Error Address Register (ADPARADDR) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-2	ERROR_ADDRESS		These bits hold the address of the first parity error generated in the ADC results' RAM. This error address is frozen from being updated until it is read by the application. In emulation mode, this address is maintained frozen even when read.
1-0	Reserved	0	Reads return 0. Writes have no effect. Reading [11:0] provides the 32-bit aligned address.

16.10.65 ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL)

Figure 16-94 and Table 16-71 describe the ADPWRDLYCTRL register.

Figure 16-94. ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL) [offset = 188h]

31	10	9	0
Reserved		PWRUP_DLY	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-71. ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	PWRUP_DLY		This register defines the number of VCLK cycles that the ADC state machine has to wait after releasing the ADC core from power down before starting a new conversion. Refer to Section 16.7.3 for more details.

16.10.66 ADC Event Group Channel Selection Mode Control Register (ADEVCHNSELMODECTRL)

Figure 16-95 and Table 16-72 describe the ADEVCHNSELMODECTRL register.

Figure 16-95. ADC Event Group Channel Selection Mode Control Register (ADEVCHNSELMODECTRL) [offset = 190h]

31	4	3	0
Reserved		EV_ENH_CHNSEL_MODE_ENABLE	
R-0		R/W-5h	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-72. ADC Event Group Channel Selection Mode Control Register (ADEVCHNSELMODECTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EV_ENH_CHNSEL_MODE_ENABLE		Enable enhanced channel selection mode for Event group. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode.
		5h	Read: Indicates that the enhanced channel selection mode for Event group is not enabled. The default sequential channel selection mode is used for Event group conversions. Write: Disables the enhanced channel selection mode for Event group and enables the sequential channel selection mode.
		Ah	Read: Indicates that the enhanced channel selection mode for Event group is enabled. Write: Enables the enhanced channel selection mode for Event group.
		All other values	Writing any value other than 5h or Ah to this field has no effect on the selected channel selection mode for the Event group, and the ADC module continues to use the channel selection mode that was previously programmed channel selection mode.

16.10.67 ADC Group1 Channel Selection Mode Control Register (ADG1CHNSELMODECTRL)

Figure 16-96 and Table 16-73 describe the ADG1CHNSELMODECTRL register.

Figure 16-96. ADC Group1 Channel Selection Mode Control Register (ADG1CHNSELMODECTRL) [offset = 194h]

31	4	3	0
Reserved		G1_ENH_CHNSEL_MODE_ENABLE	
R-0		R/W-5h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-73. ADC Group1 Channel Selection Mode Control Register (ADG1CHNSELMODECTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	G1_ENH_CHNSEL_MODE_ENABLE	5h	Enable enhanced channel selection mode for Group1. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode. Read: Indicates that the enhanced channel selection mode for Group1 is not enabled. The default sequential channel selection mode is used for Group1 conversions. Write: Disables the enhanced channel selection mode for Group1 and enables the sequential channel selection mode.
		Ah	Read: Indicates that the enhanced channel selection mode for Group1 is enabled. Write: Enables the enhanced channel selection mode for Group1.
		All other values	Writing any value other than 5h or Ah to this field has no effect on the selected channel selection mode for the Group1, and the ADC module continues to use the channel selection mode that was previously programmed channel selection mode.

16.10.68 ADC Group2 Channel Selection Mode Control Register (ADG2CHNSELMODECTRL)

Figure 16-97 and Table 16-74 describe the ADG2CHNSELMODECTRL register.

Figure 16-97. ADC Group2 Channel Selection Mode Control Register (ADG1CHNSELMODECTRL) [offset = 198h]

31	4	3	0
Reserved		G2_ENH_CHNSEL_MODE_ENABLE	
R-0		R/W-5h	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-74. ADC Group2 Channel Selection Mode Control Register (ADG2CHNSELMODECTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	G2_ENH_CHNSEL_MODE_ENABLE	5h	Enable enhanced channel selection mode for Group2. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode. Read: Indicates that the enhanced channel selection mode for Group2 is not enabled. The default sequential channel selection mode is used for Group2 conversions. Write: Disables the enhanced channel selection mode for Group2 and enables the sequential channel selection mode.
		Ah	Read: Indicates that the enhanced channel selection mode for Group2 is enabled. Write: Enables the enhanced channel selection mode for Group2.
		All other values	Writing any value other than 5h or Ah to this field has no effect on the selected channel selection mode for the Group2, and the ADC module continues to use the channel selection mode that was previously programmed channel selection mode.

16.10.69 ADC Event Group Current Count Register (ADEVCURRCOUNT)

Figure 16-98 and Table 16-75 describe the ADEVCURRCOUNT register.

Figure 16-98. ADC Event Group Current Count Register (ADEVCURRCOUNT) [offset = 19Ch]

31	5	4	0
Reserved		EV_CURRENT_COUNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-75. ADC Event Group Current Count Register (ADEVCURRCOUNT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	EV_CURRENT_COUNT		CURRENT_COUNT value for the Event group conversions when enhanced channel selection mode is enabled. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode. This register resets to zero on any of the following conditions: • A peripheral reset occurs • An ADC software reset occurs via the ADC Reset Control Register (ADRSTCR) • EV_CURRENT_COUNT becomes equal to EV_MAX_COUNT • Application writes zeros to ADEVCURRCOUNT register • Event group's result RAM is reset A read from the ADEVCURRCOUNT register returns the value of the current index into the Event group's look-up table.

16.10.70 ADC Event Group Maximum Count Register (ADEVMAXCOUNT)

Figure 16-99 and Table 16-76 describe the ADEVMAXCOUNT register.

Figure 16-99. ADC Event Group Maximum Count Register (ADEVMAXCOUNT) [offset = 1A0h]

31	5	4	0
Reserved		EV_MAX_COUNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 16-76. ADC Event Group Maximum Count Register (ADEVMAXCOUNT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	EV_MAX_COUNT		MAX_COUNT value for the Event group conversions when enhanced channel selection mode is enabled. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode. It is recommended to clear the Event group's CURRENT_COUNT register (ADEVCURRCOUNT) whenever the EV_MAX_COUNT is changed.

16.10.71 ADC Group1 Current Count Register (ADG1CURRCOUNT)

Figure 16-100 and Table 16-77 describe the ADG1CURRCOUNT register.

Figure 16-100. ADC Group1 Current Count Register (ADG1CURRCOUNT) [offset = 1A4h]

31	5	4	0
Reserved		G1_CURRENT_COUNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-77. ADC Group1 Current Count Register (ADG1CURRCOUNT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	G1_CURRENT_COUNT		CURRENT_COUNT value for the Group1 conversions when enhanced channel selection mode is enabled. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode. This register resets to zero on any of the following conditions: • A peripheral reset occurs • An ADC software reset occurs via the ADC Reset Control Register (ADRSTCR) • G1_CURRENT_COUNT becomes equal to G1_MAX_COUNT • Application writes zeros to ADG1CURRCOUNT register • Group1's result RAM is reset. A read from the ADG1CURRCOUNT register returns the value of the current index into the Group1's look-up table.

16.10.72 ADC Group1 Maximum Count Register (ADG1MAXCOUNT)

Figure 16-101 and Table 16-78 describe the ADG1MAXCOUNT register.

Figure 16-101. ADC Group1 Maximum Count Register (ADG1MAXCOUNT) [offset = 1A8h]

31	5	4	0
Reserved		G1_MAX_COUNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-78. ADC Group1 Maximum Count Register (ADG1MAXCOUNT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	G1_MAX_COUNT		MAX_COUNT value for the Group1 conversions when enhanced channel selection mode is enabled. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode. It is recommended to clear the Group1's CURRENT_COUNT register (ADG1CURRCOUNT) whenever the G1_MAX_COUNT is changed.

16.10.73 ADC Group2 Current Count Register (ADG2CURRCOUNT)

Figure 16-102 and Table 16-79 describe the ADG2CURRCOUNT register.

Figure 16-102. ADC Group2 Current Count Register (ADG2CURRCOUNT) [offset = 1ACh]

31	5	4	0
Reserved		G2_CURRENT_COUNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-79. ADC Group2 Current Count Register (ADG2CURRCOUNT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	G2_CURRENT_COUNT		CURRENT_COUNT value for the Group2 conversions when enhanced channel selection mode is enabled. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode. This register resets to zero on any of the following conditions: • A peripheral reset occurs • An ADC software reset occurs via the ADC Reset Control Register (ADRSTCR) • G2_CURRENT_COUNT becomes equal to G2_MAX_COUNT • Application writes zeros to ADG2CURRCOUNT register • Group2's result RAM is reset. A read from the ADG2CURRCOUNT register returns the value of the current index into the Group2's look-up table.

16.10.74 ADC Group2 Maximum Count Register (ADG2MAXCOUNT)

Figure 16-103 and Table 16-80 describe the ADG2MAXCOUNT register.

Figure 16-103. ADC Group2 Maximum Count Register (ADG2MAXCOUNT) [offset = 1B0h]

31	5	4	0
Reserved		G2_MAX_COUNT	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16-80. ADC Group2 Maximum Count Register (ADG2MAXCOUNT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	G2_MAX_COUNT		MAX_COUNT value for the Group2 conversions when enhanced channel selection mode is enabled. Refer to Section 16.4.2.2 for a description of the enhanced channel selection mode. It is recommended to clear the Group2's CURRENT_COUNT register (ADG2CURRCOUNT) whenever the G2_MAX_COUNT is changed.

High-End Timer (N2HET) Module

This chapter provides a general description of the High-End Timer (N2HET). The N2HET is a software-controlled timer with a dedicated specialized timer micromachine and a set of 30 instructions. The N2HET micromachine is connected to a port of up to 32 input/output (I/O) pins.

Topic **Page**

17.1 Features

- Programmable timer for input and output timing functions
- Reduced instruction set (30 instructions) for dedicated time and angle functions
- Up to maximum of 128 96-bit words of instruction RAM protected by parity. Check your datasheet for the actual number of words implemented.
- User defined configuration of 25-bit virtual counters for timer, event counters and angle counters
- 7-bit hardware counters for each pin allow up to 32-bit resolution in conjunction with the 25-bit virtual counters
- Up to 32 pins usable for input signal measurements or output signal generation
- Programmable suppression filter for each input pin with adjustable suppression window
- Low CPU overhead and interrupt load
- Efficient data transfer to or from the CPU memory with a dedicated High-End-Timer Transfer Unit (HTU)
- Diagnostic capabilities with different loopback mechanisms and pin status read back functionality
- Hardware Angle Generator (HWAG)

17.1.1 Overview

The N2HET is a fifth-generation Texas Instruments (TI) advanced intelligent timer module. It provides an enhanced feature set compared to previous generations.

This timer module provides sophisticated timing functions for real-time applications such as engine management or motor control. The high resolution hardware channels allow greater accuracy for widely used timing functions such as period and pulse measurements, output compare, and PWMs.

The reduced instruction set, based mostly on very simple, but comprehensive instructions, improves the definition and development cycle time of an application and its derivatives. The N2HET breakpoint feature, combined with various stop capabilities, makes the N2HET software application easy to debug.

17.1.1.1 Major Advantages

In addition to classic time functions such as input capture or multiple PWMs, higher-level time functions can be easily implemented in the timer program main loop. Higher-level time functions include angle driven wave forms, angle- and time-driven pulses, and input pulse width modulation (PWM) duty cycle measurement.

Because of these high-level functions, data exchanges with the CPU are limited to the fundamental parameters of the application (periods, pulse widths, angle values, and so on); and the real-time constraints for parameter communication are dramatically minimized; for example, few interrupts are required and asynchronous parameter updates are allowed.

The reduced instruction set and simple execution flow control make it simple and easy to develop and modify programs. Simple algorithms can embed the entire flow control inside the N2HET program itself. More complex algorithms can take advantage of the CPU access to the N2HET RAM. With this, the CPU program can make calculations and can modify the timer program flow by changing the data and control fields of the N2HET RAM. CPU access to the N2HET RAM also improves the debug and development of timer programs. The CPU program can stop the N2HET and view the contents of the program, control, and data fields that reside in the N2HET RAM.

Finally, the modular structure provides maximum flexibility to address a wide range of applications. The timer resolution can be selected from two cascaded prescalers to adjust the loop resolution and HR clocks. The 32 I/O pins can provide any combination of input, period or pulse capture, and output compare, including high resolution for each channel.

17.1.1.2 Timer Module Structure and Execution

The timer consists of a specialized micromachine that operates a reduced instruction set. Two 25-bit registers and three 32-bit registers are available to manipulate information such as time, event counts, and angle values. System performance is improved by a wide instruction format (96 bits) that allows the N2HET to fetch the instructional operation code and data in one system cycle, thus increasing the speed at which data can be processed. The typical operations performed in the ALU are additions (count), compares, and magnitude compares (higher or same).

Each instruction is made up of a 32-bit program field, a 32-bit control field and a 32-bit data field. The N2HET execution unit fetches the complete 96-bit instruction in one cycle and executes it. All instructions include a 9-bit field for specifying the address of the next instruction to be executed. Some instructions also include a 9-bit conditional address, which is used as the next address whenever a particular condition is true. This makes controlling the flow of an N2HET program inexpensive; in many cases a separate branch instruction is not required.

The interface to the host CPU is based on both communication memory and control registers. The communication memory includes timer instructions (program and data). This memory is typically initialized by the CPU after reset before the timer starts execution. Once the timer program is loaded into the memory, the CPU starts the timer execution, and typically data parameters are then read or written into the timer memory. The control registers include bits for selecting timer clock, configuring I/O pins, and controlling the timer module.

The programmer implements timer functions by combining instructions in specific sequences. For instance, a single count (CNT) instruction implements a timer. A simple PWM generator can be implemented with a two instruction sequence: CNT and compare (ECMP or MCMP). A complex time function may include many instructions in the sequence. The total timer program is a set of instructions executed sequentially, one after the other. Reaching the end, the program must roll to the first instruction so that it behaves as a loop. The time for a loop to execute is referred to as a *loop resolution clock cycle* or *loop resolution period (LRP).* When the N2HET rolls over to the first instruction, the timer waits for the loop resolution clock to restart the execution of the loop to ensure that only one loop is executed for each loop resolution clock.

The longest path through an N2HET program must be completed within the loop resolution clock (LRP). Otherwise, the program will execute unpredictably because some instructions will not be executed each time through the loop. This effect creates a strong link between the accuracy of the timer functions and the number of functions (the number of instructions) the timer can perform. High resolution (HR) hardware timer extensions are available for each of the N2HET pins to help overcome this limitation.

The high resolution hardware timers operate from the *high resolution clock*, which may be configured for frequency multiples between 2 and 128 times the loop resolution clock frequency. This extending the resolution of timer events and measurements well beyond what is possible with only loop resolution instructions.

Most of the commonly used N2HET instructions can operate either at loop resolution or high resolution; with the restriction that for each pin at most one high resolution instruction can be executed per loop resolution period.

Certain instructions (MOV32, ADM32, ...) can modify the data fields of other instructions. This feature enables the N2HET program to implement double buffering on capture and compare functions. For example, an ECMP compare instruction can be followed by a MOV32 instruction that is conditionally executed when the ECMP instruction matches. The host CPU can update the next compare value by writing asynchronously to the data field of the MOV32 instruction instead of writing directly to the data field of the ECMP instruction. The copy from the buffer (MOV32 data field) to the compare register (ECMP data field) will occur when the MOV32 instruction is actually executed which occurs after the ECMP instruction matches its current compare value. This is the same behavior as one would expect from a double buffered hardware compare register.

Other instructions (MOV64, RADM64) can modify both the control and data fields of other instructions. This allows the N2HET to implement toggle functionality. For example, an ECMP instruction can be followed by a pair of MOV64 instructions. The MOV64 instruction updates the data field of the ECMP instruction to implement the double buffering behavior. But it also updates the control field of the ECMP instruction which allows it to change things like pin action and the conditional address. If one MOV64 instruction configures the ECMP pin action to SET while the second changes it to CLEAR, and the two MOV64 instructions update the conditional address to point to each other, then a single ECMP instruction can be used to toggle a pin each time the compare match occurs.

17.1.1.3 Performance

Most instructions execute in one cycle, but a few take two or three cycles.

The N2HET can generate many complex output waveforms without CPU interrupts. Where special algorithms are needed following a specific event (for example, missing teeth or a short/long input signal), a minimal number of interrupts to the CPU are needed freeing the CPU for other tasks.

17.1.1.4 N2HET Compared to NHET

N2HET Enhancements from NHET include:

- Eight new instructions: ADD, ADC, SUB, SBB, AND, OR, XOR, RCNT
- Full set of ALU flags Carry (C), Negative (N), Zero (Z), Overflow (V)
- Branch instruction (BR) extended to support signed and unsigned arithmetic comparison conditions
- Two additional 32-bit temporary working registers R, S
- New HETAND register for AND-Sharing of High Resolution structure between pairs of pins
- Improved high resolution PCNT instruction

17.1.1.5 NHET and N2HET Compared to HET

Compared to the HET module, the N2HET contains all of the enhancements described in Section 17.1.1.4 plus the following additional enhancements:

- New Interrupt Enable Set and Clear registers
- Capability to generate requests to the HET Transfer Unit (HTU) including new Request Enable Set and Clear registers
- NHET RAM parity error detection
- Suppression filters for each of the 32 I/O channel and control register to configure the limiting frequency and counter clock
- Enhanced edge detection hardware that does not rely on the previous bit field in the control word of the NHET instruction
- The next, conditional and remote addresses are extended from 8 to 9 bits
- The loop resolution data fields are extended from 20 to 25 bits
- The high resolution data fields are extended from 5 to 7 bits
- Instructions with an adequate condition are able to specify the number of the request line, which triggers the HET Transfer Unit (HTU)
- The CNT instruction provides a bit, which allows to configure either an equal comparison or a greater or equal comparison when comparing the selected register value with the Max-value
- The MOV32 instruction provides a new bit. If set to one the MOV32 will only perform the move, when the Z-flag is set. If set to zero the MOV32 will perform the move whenever it is executed (independent on the state of the Z-flag)
- There is a new instruction WCAPE, which is a combination of a time stamp and an edge counter
- New Open Drain, Pull Disable, and Pull Select registers

17.1.1.6 Instructions Features

The N2HET has the following instructions features:

- N2HET uses a RISC-based specialized timer micromachine to carry out a set of 30 instructions
- Instructions are implemented in a Very Long Instruction Word (VLIW) format (96-bits wide)
- The N2HET program execution is self-driven by external or internal events, branching to special routines based on input edges or output compares
- Instructions point to the next instruction executed, eliminating the need for a program counter
- Several instructions can change the program flow based on internal or external conditions

17.1.1.7 Program Usage

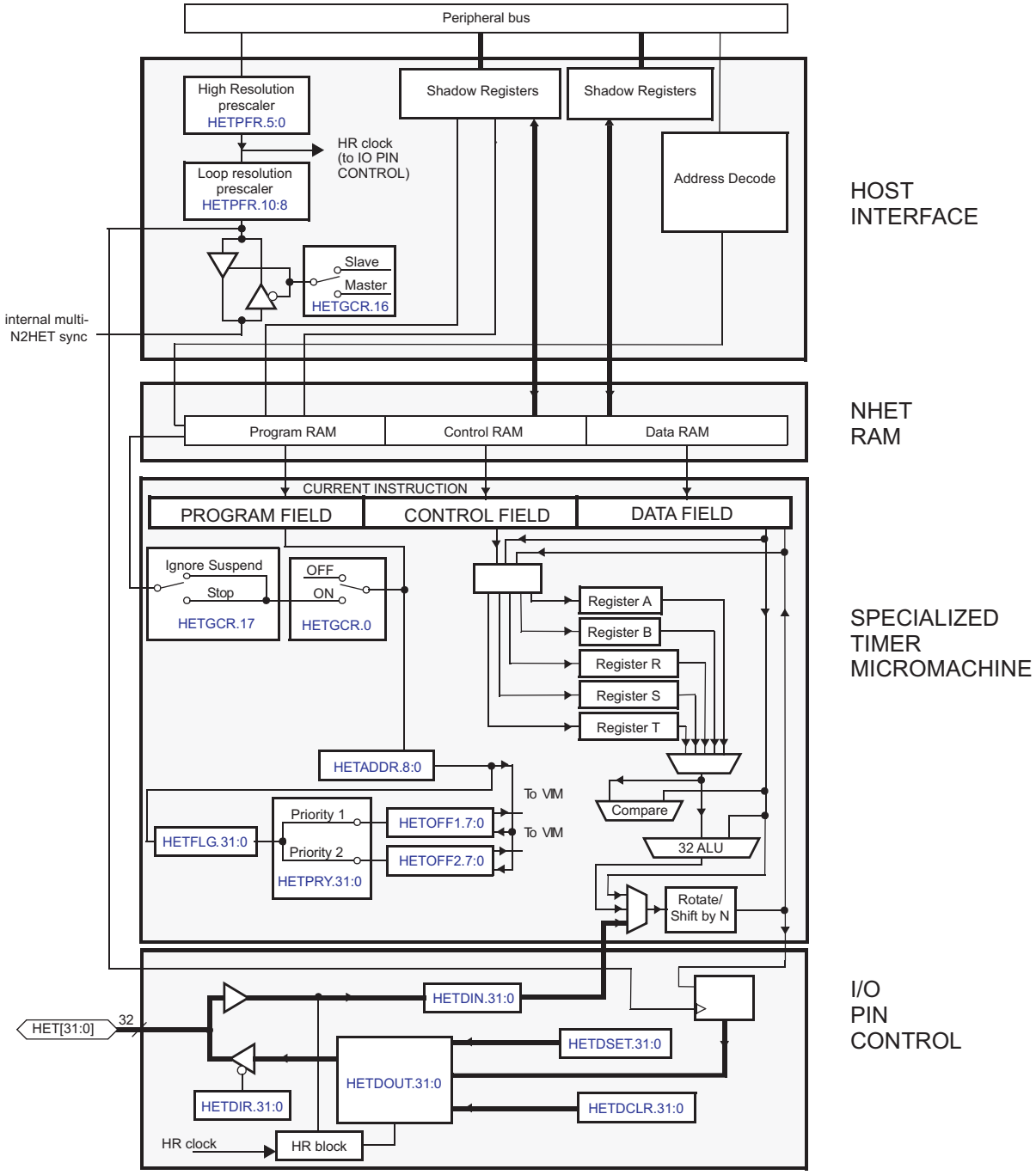
The N2HET instructions/program can be assembled with the N2HET assembler. The assembler generates a C-structure which can be included into the main application program. The application has to copy the content of the structure into the N2HET RAM, set up necessary registers and start the N2HET program execution. In addition to the C-structure, the assembler generates also a header file which makes it easy for the main application to access the different instructions and change for example the duty cycle of a PWM or read out the captured value of a specific signal edge.

17.1.2 Block Diagram

The N2HET module (see Figure 17-1) comprises four separate components:

- Host interface
- N2HET RAM
- Specialized timer micromachine
- I/O control (the N2HET is attached to an I/O port of up to 32 pins)

Figure 17-1. N2HET Block Diagram



17.2 N2HET Functional Description

The N2HET contains RAM into which N2HET code is loaded. The N2HET code is run by the specialized timer micromachine. The host interface and I/O control provide an interface to the CPU and external pins respectively.

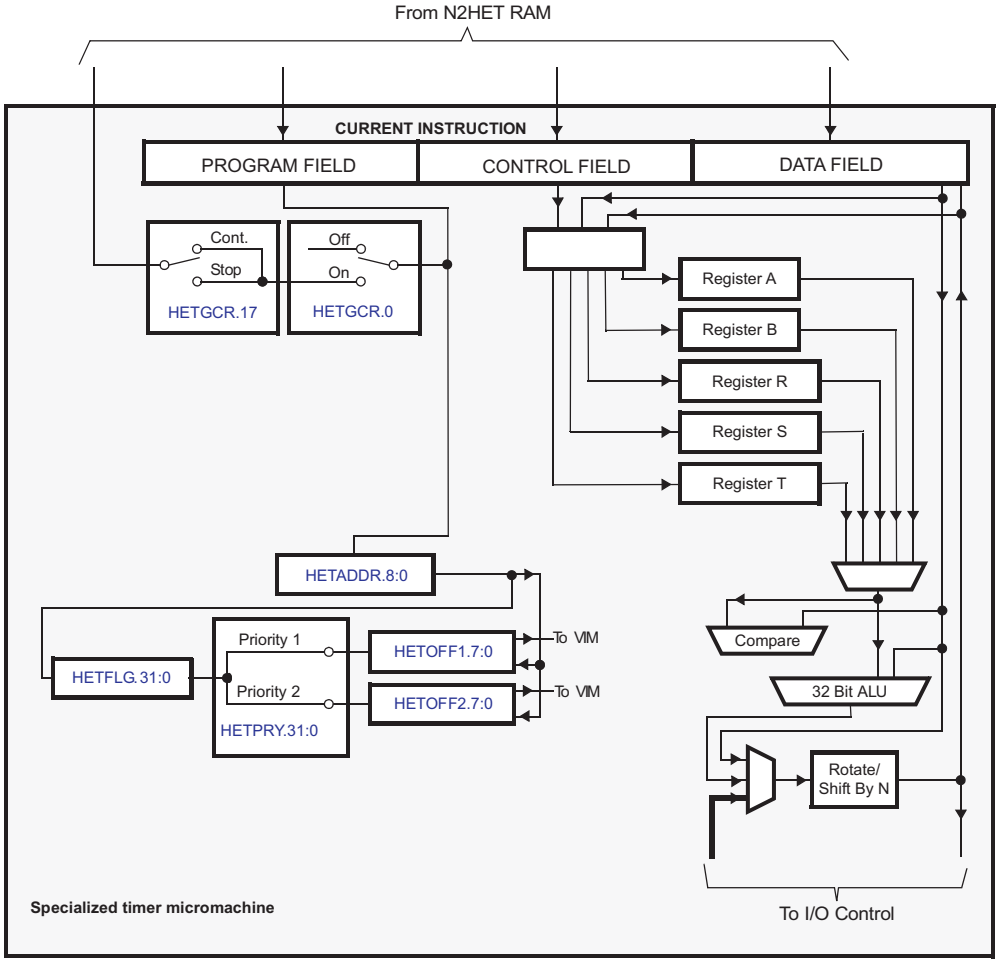
17.2.1 Specialized Timer Micromachine

The N2HET has its own instruction set, detailed in Section 17.5.1. The timer micromachine reads each instruction from the N2HET RAM. The program and control fields contain the instructions for how the specialized timer micromachine executes the command. For most instructions, the data field stores the information that needs to be manipulated.

The specialized timer micromachine executes the instructions stored in the N2HET RAM sequentially. The N2HET program execution is self-driven by external or internal events. This means that input edges or output compares may force the program to branch to special routines using a conditional address.

Figure 17-2 shows some of the major operations that the N2HET can carry out, namely compares, captures, angle functions, additions, and shifts. The N2HET contains five registers (A, B, R, S, and T) used to hold compare or counter values and are used by the N2HET instructions. Data may be taken from the registers or the data field for manipulation; likewise, the data may be returned to the registers or the data field.

Figure 17-2. Specialized Timer Micromachine



17.2.1.1 Time Slots and Resolution Loop

Each instruction requires a specific number of cycles or time slots to execute. The resolution specified in the prescaler bitfields determines the timer accuracy. All input captures, event counts, and output compares are executed once in each resolution loop. HR captures and compares are possible (up to N2HET clock accuracy) on the HR I/O pins. For more information about the HR I/O structure, see Section 17.2.5.

17.2.1.2 Program Loop Time

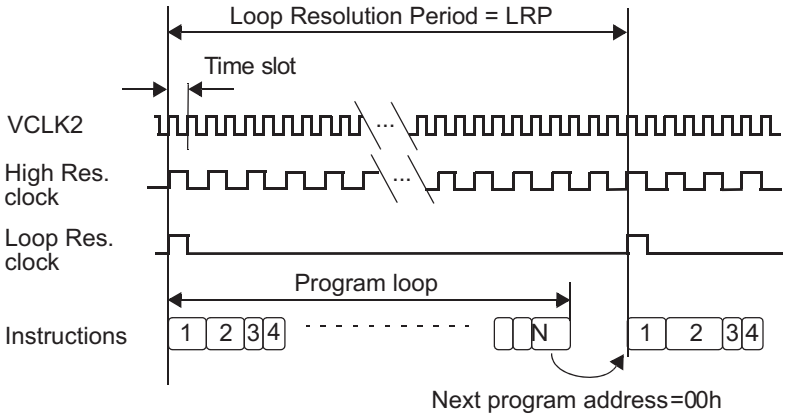
The program loop time is the sum of all cycles used for instruction execution. This time may vary from one loop to another if the N2HET program includes conditionally executed instructions.

The timer program restarts on every resolution loop. The start address is fixed at N2HET RAM address 00h. The longest path through a program must fit within one loop resolution period to guarantee complete accuracy.

The last instruction of a program must branch back to the fixed start address (next program address= 00h). When an N2HET program branches back to address 00h before the end of a loop resolution period, the N2HET detects this and pauses instruction execution until the beginning of the next loop resolution period.

The timing diagram below illustrates the program flow execution.

Figure 17-3. Program Flow Timings



17.2.1.3 Instruction Execution Sequence

The execution of a N2HET program begins with the first occurrence of the loop resolution clock, after the N2HET is turned on. At the first and subsequent occurrences of the loop resolution, the instruction at location address 00h is prefetched. The program execution begins at the occurrence of the loop resolution clock and continues executing the instructions until the program branches to 00h location. The instruction is prefetched at location 00h and execution flag is reset. The N2HET pauses instruction execution until the occurrence of the loop resolution clock and resumes normal execution.

N2HET programs must be written so that they complete execution and return to address 00h before the occurrence of the next loop resolution clock. If the N2HET program exceeds this execution time limit, then a program overflow condition occurs as described in Section 17.2.1.4.

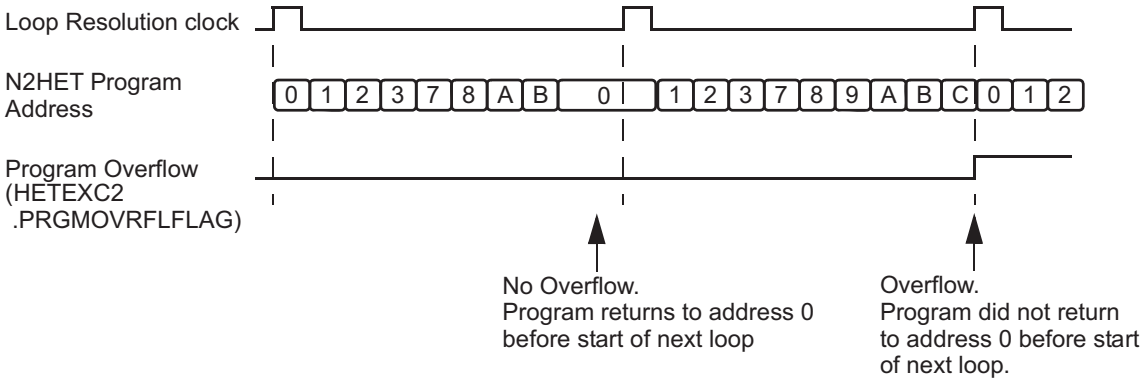
17.2.1.4 Program Overflow Condition

If the number of time slots used in a program loop exceeds the number available time slots in one loop resolution, the timer sets the program overflow interrupt flag located in the HETEXC2 register. To maintain synchronization of the I/Os, this condition should never be allowed to occur in a normal operation. The HETEXC2.PRGMOVRFLFLAG flag provides a mechanism for checking that the condition does not occur during the debug and validation phases.

As Figure 17-4 illustrates, when a program overflow occurs, the currently executing N2HET program sequence is interrupted and restarted at N2HETaddress 0 for the beginning of the next loop resolution clock period. Also, HETEXC2.PRGMOVRFLFLAG is set.

If the instruction that caused the overflow (instruction at address 0xC in Figure 17-4) has any pin actions selected, these pin actions will not be performed. However other actions of the instruction including register and RAM updates will still be performed.

Figure 17-4. Use of the Overflow Interrupt Flag (HETEXC2)



17.2.1.5 Architectural Restrictions on N2HET Programs

Certain architectural restrictions apply to N2HET programs:

1. The size of an N2HET program must be greater than one instruction.
2. An extra wait state is incurred by any instruction that modifies a field in the next instruction to be executed.
3. Only one instruction (using high resolution) is allowed per high resolution pin.
4. Consecutive break points are not supported. Instructions with break points must have at least a distance of two instructions (for example, at addresses 1, 3, 5, 7, and so on, assuming the program executes linearly)

NOTE: While it would be unusual to code an N2HET program that is only one instruction long, it is trivial to modify such a program to meet the requirement of restriction 1. Simply add a second instruction to the program, which may be a simple branch to zero.

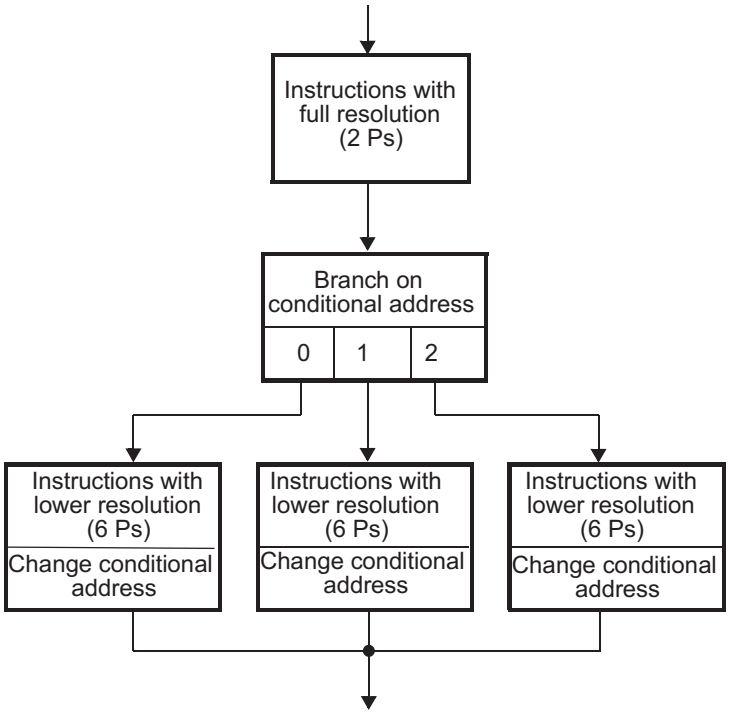
To enforce restriction 3, the high resolution pin structures respond only to the first instruction that is executed matching their pin number with hr_lr=HIGH, regardless of whether or not the en_pin_action field is ON. Subsequent instructions are ignored by the high resolution pin structure for the remainder of the loop resolution period.

17.2.1.6 Multi-Resolution Scheme

The N2HET has the capability to virtually extend the counter width by executing instructions only once every N loop resolution periods. This decreases the timer resolution, but extends the counter range which may be useful when generating or measuring slow signals. Figure 17-5 illustrates how a multi-resolution scheme may be implemented in an N2HET program. An unconditional Branch instruction and an index sequence, using a MOV64 instruction in each low resolution loop, is required to control this particular program flow.

NOTE: HR instructions must be placed in the main (full resolution) loop to ensure proper operation.

Figure 17-5. Multi-Resolution Operation Flow Example



17.2.1.7 Debug Capability

The N2HET supports breakpoints to allow you to more easily debug your N2HET program. Figure 17-6 provides an illustration of the breakpoint mechanism.

The steps to enable an N2HET breakpoint are:

1. Make sure the device nTRST pin is high, since N2HET breakpoints are disabled whenever this pin is low. (Normally this is handled automatically when a JTAG debugger is attached).
2. Attach a JTAG debugger and connect to the device that has been already programmed with the N2HET code that needs to debugged. (downloading to on-chip flash is outside the scope of this section).
3. Execute the CPU program at least until the point where the N2HET program RAM has been initialized by the CPU.
4. Open a memory window in the N2HET registers.
5. Make sure NHETEXC2.DEBUGSTATUSFLAG is cleared.
6. Open a memory window on the N2HET RAM
7. Set bit 22 in the program field of the instruction(s) on which you wish to break. Note that this instruction will be executed **before** the N2HET is halted - slightly different from how CPU breakpoints behave.
8. Make sure the CPU and N2HET are running, if they are halted then restart the CPU through the JTAG emulator (N2HET will start when the CPU starts).
9. Both the CPU and N2HET will halt when breakpoint is reached.

When the N2HET is halted, its state machines are frozen but all of the N2HET control registers can be accessed through the JTAG emulator interface.

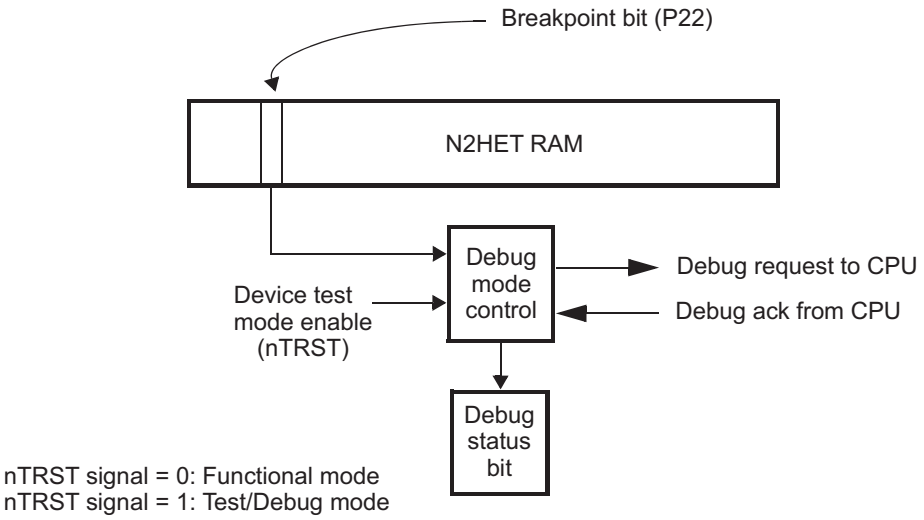
The current N2HET instruction address can be inspected by reading the HETADDR register; this should be pointing to the instruction that caused the breakpoint.

The N2HET internal working registers (A,B,R,S,T) are not directly visible through the JTAG emulator interface. If the content of these registers needs to be inspected, it is best to add an instruction like MOV32 which copies the register value to the N2HET RAM. This RAM location can be inspected when the N2HET halts.

To restart execution of both the CPU and the N2HET from the halted state:

1. Clear NHETEXC2.DEBUGSTATUSFLAG.
2. Clear bit 22 in the program field of the instruction on which the breakpoint was reached.
3. Restart the CPU through the normal JTAG emulator procedure ('Run' or 'Go'). The N2HET will automatically start executing when it sees that the CPU has exited the debug state.

Figure 17-6. Debug Control Configuration



NOTE: Consecutive break points are not supported. Instructions with break points must have at least a distance of two instructions (for example, at N2HET addresses 1, 3, 5, 7, and so on)

17.2.2 N2HET RAM Organization

The N2HET RAM is organized into two sections. The first contains the N2HET program itself. The second contains parity protection bits for the N2HET program.

Each N2HET instruction is 96-bits wide but aligned to a 128-bit boundary. Instructions consist of three 32-bit fields: Program, Control, and Data. Instructions are separated by a fourth unimplemented address to force alignment to 128-bit boundaries.

The integrity of the N2HET program can be protected by Parity. Parity protection is enabled through the N2HET Parity Control Register (HETPCR).

Table 17-1 shows the base addresses for N2HET RAM and N2HET Parity RAM.

Table 17-1. N2HET RAM Base Addresses

N2HET Base Address	Memory
0xFF46 0000	N2HET Instruction RAM (Program/Control/Data)
0xFF46 2000	N2HET Parity RAM

17.2.2.1 N2HET RAM Banking

Because the CPU must make updates to the N2HET RAM while the N2HET is executing, for example to update the duty cycle value of a PWM, it is important to understand how the N2HET RAM organization facilitates simultaneous accesses by both the HOST CPU and the N2HET.

The N2HET RAM is implemented as 4 banks of 96-bit wide two port RAM. This means that there a total of 8 ports available; four read and four write. Normally the N2HET will use up to two of these ports at a time. One read port is used to allow the N2HET to prefetch the next N2HET instruction while a write port may be used to update the data or control fields that have changed as a result of executing the current instruction.

N2HET accesses to its own internal RAM are given priority over accesses from an external host (CPU or HTU), this makes N2HET program execution deterministic which is a critical requirement for a timer.

Most N2HET instructions execute in a single cycle. Cases where a wait state impacts the N2HET program execution time are:

- The current N2HET instruction writes data back to the next N2HET in the execution sequence.
- The external host reads from an N2HET instruction where the automatic read-clear option is set, while the N2HET is executing from/on the same address (See Section 17.2.4.3).

Except for the case of automatic read-clear, the external host is stalled when the host and N2HET have a bank conflict. However this will typically only result in a stall of one cycle, due to the N2HET bank ordering which is organized on the N2HET Address least significant bit boundaries (See Table 17-2).

Assuming most of the N2HET program executes linearly through the N2HET Address space; if a bank conflict does exist it is usually resolved in the next cycle as the N2HET program moves to the next bank. N2HET programmers should avoid writing a program that accesses the same bank of N2HET RAM on every cycle, as this could lock the external host out of the N2HET memory completely.

Table 17-2 describes the N2HET memory map, as viewed by the N2HET as well as from the memory space of the host CPU.

Table 17-2. N2HET RAM Bank Structure

N2HET Address	Host CPU Address Space				
Instruction	Program Field Address	Control Field Address	Data Field Address	Reserved Address	N2HET RAM Bank
000h	XX0000h	XX0004h	XX0008h	XX000Ch	A
001h	XX0010h	XX0014h	XX0018h	XX001Ch	B
002h	XX0020h	XX0024h	XX0028h	XX002Ch	C
003h	XX0030h	XX0034h	XX0038h	XX003Ch	D
004h	XX0040h	XX0044h	XX0048h	XX004Ch	A
:	:	:	:	:	:
03Fh	XX03F0h	XX03F4h	XX03F8h	XX03FCh	D
040h	XX0400h	XX0404h	XX0408h	XX040Ch	A
:	:	:	:	:	:
1FFh	XX1FF0h	XX1FF4h	XX1FF8h	XX1FFCh	D

NOTE: The external host interface supports any access size for reads, but only 32-bit writes to the N2HET RAM are supported. Reserved addresses should not be accessed, the result of doing so is indeterminate.

17.2.2.2 Parity Checking

The N2HET module can detect parity errors in N2HET RAM. As described in Section 17.2.2, the N2HET allows 32-bit writes only. Therefore, N2HET RAM parity checking is implemented using one parity bit per 32-bit field in N2HET RAM.

Even or odd parity selection for N2HET parity detection can be configured in the system module. Parity calculation and checking can be enabled/disabled by a 4-bit key in HETPCR.

During a read access to the N2HET RAM, the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. The parity check is performed when the N2HET execution unit makes a read access to N2HET RAM, but also when a different master (for example, CPU or HTU) performs the read access. If any 32-bit-word fails the parity check then an error is signaled to the ESM module. The N2HET address, which generated the error is detected and is captured in HETPAR for host system debugging. The address is frozen from being updated until it is read by the bus master.

The N2HET execution unit reads the instructions, which are 96-bit wide. They contain the program-, control- and data-field whereby each is 32-bit wide. So when fetching N2HET instructions parity checking is performed on three words in parallel.

If a parity error is detected in two or more words in the same cycle then only one address (word at the lower address) is captured. The captured N2HET address is always aligned to a 32-bit word boundary.

During debug, parity checking is still performed on accesses originating from the on-chip host CPU or HTU. However, parity errors that are detected during an access initiated by the debugger are ignored.

17.2.2.3 Parity Error Detection Actions

Detection of a N2HET parity error causes the following actions:

1. An error is signaled to the ESM module.
2. The Parity Address Register (HETPAR) is loaded with the address of the faulty N2HET field.
3. N2HET execution immediately stops. (The instruction that triggered the parity error is not executed.)
4. The Turn-On/Off-Bit in the N2HET Global Configuration Register (HETGCR) is automatically cleared.
5. All N2HET internal flags are cleared.
6. All N2HET pins selected by N2HET Parity Pin Register (HETPPR) enter a predefined safe state.
7. Register HETDOUT is also updated to reflect changes in pin state due to HETPPR.

The safe state for N2HET pins selected through the HETPPR register depends on how the pin is configured in the HETDIR, HETPDR, and HETPSL registers. Table 17-3 explains how the safe state is determined.

Table 17-3. Pin Safe State Upon Parity Error Detection

Safe State	HETDIR	HETPDR	HETPSL
Drive Low	1	0	0
Drive High	1	0	1
High Impedance	1	1	x

17.2.2.4 Testing Parity Detection Logic

To test the parity detection logic, the parity RAM has to be made accessible to the CPU in order to allow a diagnostic program to insert parity errors. The control register bit HETPCR.TEST must be set in order to make the parity RAM accessible. Once HETPCR.TEST is set, the parity bits are accessible as described in Table 17-4.

Each 32-bit N2HET field has its own parity bit in the N2HET Parity RAM as shown in Table 17-4. There are no parity bits for the reserved fields, since there is no physical N2HET RAM for these fields.

Table 17-4. N2HET Parity Bit Mapping

Address N2HET1	Bits	
	[31:1]	[0]
0xFF46_2000	Reads 0, Writes have no effect	Instruction 0 Program Field Parity Bit
0xFF46_2004	Reads 0, Writes have no effect	Instruction 0 Control Field Parity Bit
0xFF46_2008	Reads 0, Writes have no effect	Instruction 0 Data Field Parity Bit
0xFF46_200C	Reads 0, Writes have no effect	Read 0
0xFF46_2010	Reads 0, Writes have no effect	Instruction 1 Program Field Parity Bit
....	...	...

17.2.2.5 Initialization of Parity RAM

After device power up, the N2HET RAM contents including the parity bits cannot be guaranteed. In order to avoid false parity failures due to the random state in which RAM powers up, the RAM has to be initialized.

Before initializing the N2HET RAM, enable the N2HET parity logic by writing to HETPCR. Then the N2HET Instruction RAM should be initialized. With parity enabled, the N2HET parity RAM will be initialized automatically by N2HET at the same time that the N2HET instruction RAM is initialized by the CPU. Note that loading the N2HET program with parity enabled is also effective.

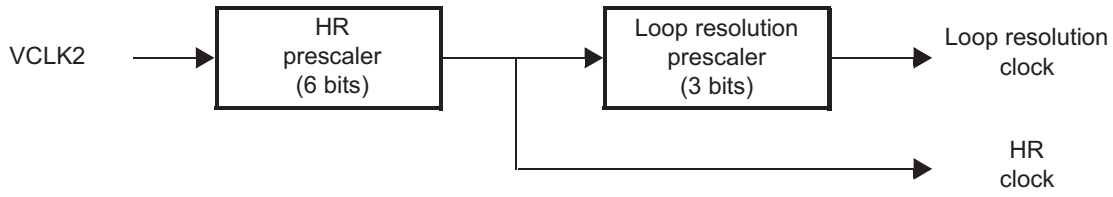
Another possibility to initialize the N2HET memory and its parity bits is, to use the system module to start the automatic initialization of all RAMs on the microcontroller. The RAMs will be initialized to ‘0’. Depending on the even/odd parity selection, the parity bit will be calculated accordingly.

17.2.3 Time Base

All N2HET timings are derived from VCLK2 (see Figure 17-7). Internally N2HET instructions execute at the VCLK2 rate; but the timer loop clock and the high-resolution hardware timer clock can be scaled down from VCLK2. Two prescalers are available to adjust the timer loop resolution clock for the program loop, and the high resolution (HR) clock for the HR I/O counters.

- **Time Slots:** The number of cycles available for instruction execution per loop. Time Slots is the number of VCLK2 cycles in a Loop Resolution Clock.
- **High Resolution Clock:** The high resolution clock is the smallest time increment with which a pin can change it’s state or can be measured in the case of input signals. A 6-bit prescaler dividing VCLK2 by a user-defined HR prescale divide rate (hr) stored in the 6-bit HR prescale factor code (HETPFR). See Table 17-5.
- **Loop Resolution Clock:** The loop resolution clock defines the timebase for executing all instructions in a N2HET program. Since instructions can be conditionally executed, the longest path through the N2HET program must fit into one loop resolution clock period (LRP).A 3-bit prescaler dividing the HR clock by a user-defined loop-resolution prescale divide rate (lr) stored in the 3-bit loop-resolution prescale factor code (HETPFR). See Table 17-5.

Figure 17-7. Prescaler Configuration



The following abbreviations and relations are used in this document:

1. hr: high resolution prescale factor (1, 2, 3, 4,..., 63, 64)
2. lr: loop resolution prescale factor (1, 2, 4, 8, 16, 32, 64,128)
3. ts: Time slots (cycles) available for instruction execution per loop. ts = hr x lr
4. HRP = high resolution clock period HRP = hr · T_{VCLK2} (ns)
5. LRP = loop resolution clock period LRP = lr · HRP (ns)

The loop resolution period (LRP) must be selected to be larger than the number of Time slots (VCLK2 cycles) required to complete the worst-case execution path through the N2HET program. Otherwise a program overflow condition may occur (see Section 17.2.1.4). Because of the relationship of time slots to the hr and lr prescalers as described in item 3 above, increasing either hr or lr increases the number of time slots available for program execution. However, lr would typically be increased first, since increasing hr results in a decrease in timer resolution since it reduces the clock to the High Resolution IO structures.

The divide rates hr and lr can be defined in the HETPFR register. Table 17-5 lists the bit field encodings for the prescale options.

Table 17-5. Prescale Factor Register Encoding

LRPFC - Loop Resolution		HRPFC - High Resolution	
HETPFR[10:8]	**Prescale Factor lr**	**HETPFR[5:0]**	**Prescale Factor hr**
000	/1	000000	/1
001	/2	000001	/2
010	/4	000010	/3
011	/8	000011	/4
100	/16	: : : : :	:
101	/32	111101	/62
110	/64	111110	/63
111	/128	111111	/64

17.2.3.1 Determining Loop Resolution

As an example, consider an application that requires high resolution of HRP = 62.5 ns, and loop resolution of LRP = 8 µs, and needs at least 250 time slots for the N2HET application program.

Assuming VCLK2 = 32 MHz, the following shows which divide-by rates and which value in the Prescale Factor Register (HETPFR) is required for the above requirements:

$$hr = 2 \rightarrow HRP = \frac{hr}{VCLK2} = \frac{2}{32MHz} = 62.5ns$$

$$lr = 128 \rightarrow lr \times HRP = 128 \times 62.5ns = 8\ \mu s$$

$$ts = hr \times lr = 2 \times 128 = 256$$

$$hr = 2, lr = 128 \rightarrow HETPFR[31:0] = 0x00000701 \quad (32)$$

In the example above, if the loop resolution period needs to decrease from 8 µs to 4 µs, then only 128 time slots will be available for program execution. The program may need to be restructured as suggested in Section 17.2.1.6.

17.2.3.2 The 7-Bit HR Data Field

The instruction execution examples of ECMP (Section 17.2.5.9), MCMP (Section 17.2.5.10), PCNT (Section 17.2.5.12), PWCNT (Section 17.2.5.11), and WCAP (Section 17.2.5.13) show that the 7-bit HR data field can generate or measure high resolution delays (HR delay) relative to the start of an LRP within one N2HET loop LRP. The last section showed that:

LRP = lr × HRP

There are lr high resolution clock periods (HRP) within the N2HET loop resolution clock period (LRP). If lr = 128 then the HR delay can range from 0 to127 HRP clocks within LRP and all 7 bits of the HR data field are needed. Instead of being limited to measuring and triggering events based on the loop resolution clock period (LRP) the HR extension allows measurements and events to be described in terms fractions of an LRP (down to 1/128 of an LRP). The only limitation is that a maximum of one HR delay can be specified per pin during each loop resolution period.

Table 17-6 shows which bits of the HR data field are not used by the high resolution IO structures if lr is less than 128. In this case the non-relevant bits (LSBs) of the HR data fields will be one of the following:

- Written as 0 for HR capture (for PCNT, WCAP)
- Or interpreted as 0 for HR compare (for ECMP, MCMP. PWCNT)

Table 17-6. Interpretation of the 7-Bit HR Data Field

Loop Resolution Prescale divide rate (lr)	Bits of the HR data field [1]							HRP Cycles delay range
	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]	
1	X	X	X	X	X	X	X	0
2	1/2	X	X	X	X	X	X	0 to 1
4	1/2	1/4	X	X	X	X	X	0 to 3
8	1/2	1/4	1/8	X	X	X	X	0 to 7
16	1/2	1/4	1/8	1/16	X	X	X	0 to 15
32	1/2	1/4	1/8	1/16	1/32	X	X	0 to 31
64	1/2	1/4	1/8	1/16	1/32	1/64	X	0 to 63
128	1/2	1/4	1/8	1/16	1/32	1/64	1/128	0 to 127

(1) X = Non-relevant bit (treated as 0)

17.2.3.2.1 Example:

Prescale Factor Register (HETPFR) = 0x0300

—> lr = 8 —> LRP = 8 · HRP

Assumption: HR data field = 0x50 = 1010000b

lr = 8 —> Bits D[3:0] are ignored —> HR delay = 101b = 5 HRPs

or by using the calculation with weight factors:

HR Delay
= lr · (D[6] · 1/2 + D[5] · 1/4 + D[4] · 1/8 + D[3] · 1/16 + D[2] · 1/32 + D[1] · 1/64 + D[0] · 1/128)
= 8 · (1 · 1/2 + 0 · 1/4 + 1 · 1/8 + 0 · 1/16 + 0 · 1/32 + 0 · 1/64 + 0 · 1/128)
= 5 HRPs

17.2.4 Host Interface

The host interface controls all communications between timer-RAM and masters accessing the N2HET RAM. It includes following components:

17.2.4.1 Host Accesses to N2HET RAM

The host interface supports the following types of accesses to N2HET RAM:

- Read accesses of 8, 16, or 32 bits
- Read accesses of 64-bits that follow the shadow register sequence described in Section 17.2.4.2.
- Write accesses of 32 bits

Writes of 8 or 16 bits to N2HET RAM by an external host are not supported.

17.2.4.2 64-bit Read Access

The consecutive read of a control field CF(n) and a data field DF(n) of the same instruction (n) performed by the same master (for example, CPU or HTU) is always done as a simultaneous 64-bit read access. This means that at the same time CF(n) is read, DF(n) is loaded in a shadow register. So the second access will read DF(n) from the shadow register instead of the N2HET RAM.

In general a 64-bit read access of one master could be interrupted by a 64-bit read access of another master. A total of three shadow registers are available. Therefore up to three masters can perform 64-bit reads in an interleaved manner (Master1 CF, Master2 CF, Master3 CF, Master1 DF, Master2 DF, Master3 DF).

If all three shadow registers are activated and a 4th master performs a CF or DF read it will result in an address error and the RAM access will not happen. Other access types by a fourth master (reads from the PF field or writes to any of the fields) will occur because these access types do not require an available shadow register resource to complete.

17.2.4.3 Automatic Read Clear Feature

The N2HET provides a feature allowing to automatically clear the data field immediately after the data field is read by the external host CPU (or HTU). This feature is implemented via the *control bit*, which is located in the control field (bit C26). This is a static bit that can be used by any instruction, and specified in the N2HET program by adding the option (control = ON) to the N2HET instruction. The automatic read clear feature works for both 32 and 64 bit reads that follow the sequence described in Section 17.2.4.2.

When the host CPU reads the data field of that instruction, the current data value is returned to the host CPU but the field is cleared automatically as a side effect of the read. In case the master reads data from an instruction currently executing, any new capture result is stored and this takes priority over the automatic read clear feature, so that the new capture result is not lost.

As an example of where the automatic read clear feature is useful, consider the PCNT instruction. If this instruction is configured for automatic read clear, then when the host CPU reads the PCNT data field it will be cleared automatically. The host CPU can then poll the PCNT data field again, and as long as the field returns a value of zero the host CPU program knows a new capture event has not occurred. If the data field were not cleared, it would be impossible for the host CPU to determine whether the data field holds data from the previous capture event, or if it happens to be data from a new capture event with the same value.

17.2.4.4 Emulation Mode

Emulation mode, used by the software debugger, is specified in the global configuration register. When the host CPU debugger hits a breakpoint, the CPU sends a suspend signal to the modules. Two modes of operation are provided: suspend and ignore suspend.

- Suspend

When a suspend is issued, the timer operation stops at the end of the current timer instruction. However, the CPU accesses to the timer RAM or control registers are freely executed.

- Ignore suspend

The timer RAM ignores the suspend signal and operates real time as normal.

17.2.4.5 Power-Down

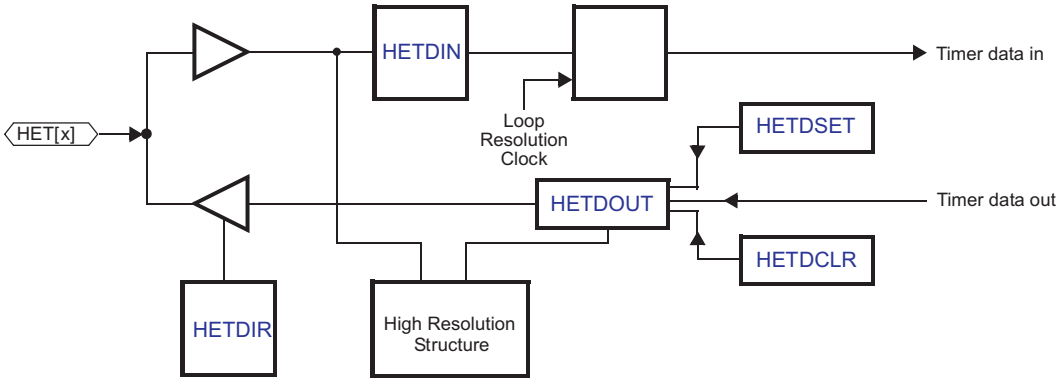
After setting the turn-off bit in the Global Configuration Register (HETGCR), it is required to delay until the end of the timer program loop before putting the N2HET in power-down mode. This can be done by waiting until the N2HET Current Address (HETADDR) becomes zero, before disabling the N2HET clock source in the device's Global Clock Module (GCM).

17.2.5 I/O Control

The N2HET has up to 32 pins. Refer to device-specific data manual for information concerning the number of N2HETIO available. All of the N2HET pins available are programmable as either inputs or outputs.

These 32 I/Os have an identical structure connected to pins HET[31] to HET[0]. See Figure 17-8 for an illustration of the I/O control. In addition all 32 I/Os have a special HR structure based on the HR clock. This structure allows any N2HET instruction to use any of these I/Os with an accuracy of either loop resolution or high resolution accuracy.

Figure 17-8. I/O Control



Pins N2HET [31] to N2HET [0] can be used by the CPU as general purpose inputs or outputs using the N2HET Data Input Register (HETDIN) for reading and N2HET Data Output Register (HETDOUT), N2HET Data Set Register (HETDSET) or N2HET Data Clear Register (HETDCLR) for writing, depending on the type of action to perform. The N2HET pins used as general purpose inputs are sampled on each VCLK2 period.

17.2.5.1 Using General Purpose I/O Data Set and Clear Registers

The N2HET Data Clear Register (HETDCLR) and N2HET Data Set Register (HETDSET) can be used to minimize the number of accesses to the peripheral to modify the output register and output pins. When the application needs to set or to reset some N2HET pins pins without changing the value of the others pins, the first possibility is to read N2HET Data Output Register (HETDOUT), modify the content (AND, OR, and so on), and write the result into N2HET Data Output Register (HETDOUT). However, this read-modify-write sequence could be interrupted by a different function modifying the same register which will result in a data coherency problem.

Using the N2HET Data Set Register (HETDSET) or N2HET Data Clear Register (HETDCLR), the application program must write the mask value (same mask value for the first option) to the register to set or reset the desired pins. Any bits written as 0 to HETDSET and HETDCLR are left unchanged, which avoids the possible coherency problem of the read-modify-write approach.

```
Coding Example (C program): Set pins using the 2 methods.

unsigned int MASK;                              /* Variable that content the bit mask  */
volatile unsigned int *HETDOUT,*HETDSET;        /* Pointer to HET registers            */
...
*HETDOUT = *HETDOUT | MASK;                     /* Read-modify-write of HETDOUT        */
*HETDSET = MASK;                                /* Set the pin without reading HETDOUT */
```

17.2.5.2 Loop Resolution Structure

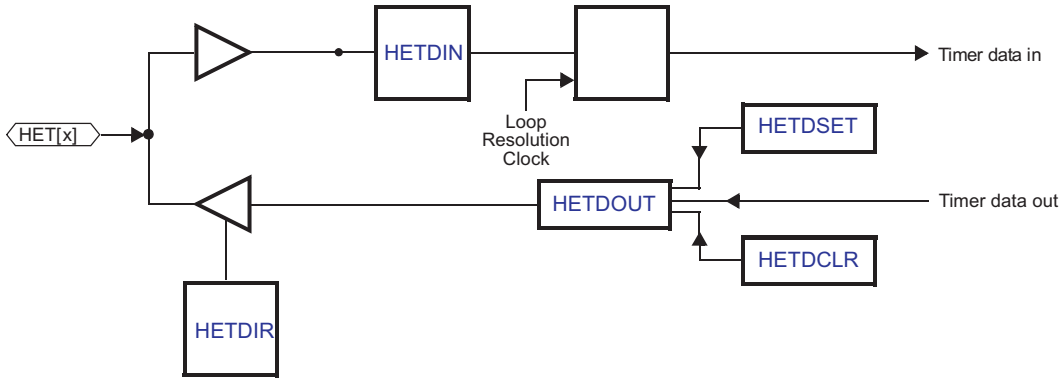
The N2HET uses the pins N2HET [31:0] as input and/or output by the way of the instruction set. Actually, each pin could monitor the N2HET program or could be monitored by the N2HET program. By using the I/O register of the N2HET, the CPU is able to interact with the N2HET program flow.

When an action (set or reset) is taken on a pin by the N2HET program, the N2HET will modify the pin at the rising edge of the next loop resolution clock.

When an event occurs on a N2HET I/O pin, it is taken into account at the next rising edge of the loop resolution clock.

The structure of each pin is shown in Figure 17-9.

Figure 17-9. N2HET Loop Resolution Structure for Each Bit

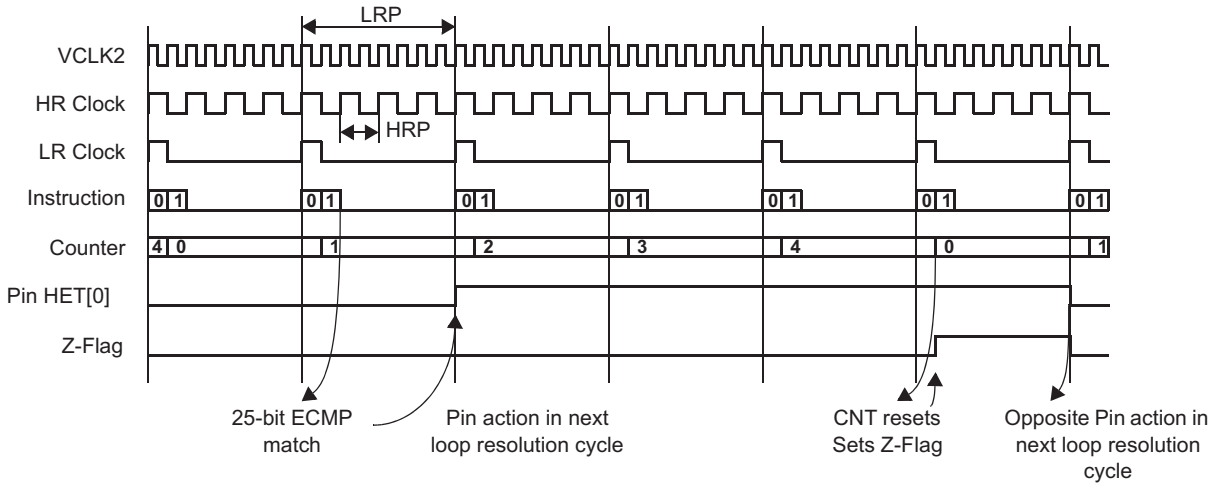


The example in Figure 17-10 shows a simple PWM generation with loop resolution accuracy. The corresponding program is:

```
HETPFR[31:0] register = 0x201 --> lr=4 and hr=2 --> ts = 8

N2HET Program:

L00  CNT   { next= L01, reg=A, irq=OFF, max = 4 }
L01  ECMP  { next= L00, cond_addr= L00, hr_lr=LOW, en_pin_action=ON, pin=0,
             action=PULSEHI, reg=A, irq=OFF, data= 1, hr_data = 0x0 }

; 25 bit compare value is 1 and the 7-bit HR compare value is 0
```

The CNT and ECMP instructions are executed once each loop resolution cycle. When the CNT instruction is executed, the specified register (A) and the CNT instruction data field are both incremented by one. Next the ECMP is executed and the data field of the ECMP is compared with the specified register (A). If both values match, then the pin action (PULSEHI in this case) will be performed in the next loop resolution cycle. The CNT continues incrementing each loop resolution cycle. When the data field overflows (max + 1), then the Z-flag is set by the CNT instruction. In the next loop resolution cycle, the Z-flag is evaluated and the opposite pin action is performed if it is set. The Z-flag will only be active for one loop resolution cycle.

Figure 17-10. Loop Resolution Instruction Execution Example



17.2.5.3 High Resolution Structure

All 32 I/Os provide the HR structure based on the HR clock. The HR clock frequency is programmed through the Prescale Factor Register (HETPFR). In addition to the standard I/O structure, all pins have HR hardware so that these pins can be used as HR input captures (using the HR instructions PCNT or WCAP) or HR output compares (using the HR instructions ECMP, MCMP, or PWCNT).

All five HR instructions (PCNT, WCAP, ECMP, MCMP, and PWCNT) have a dedicated hr_lr bit (high resolution/low resolution; program field bit 8) allowing operation either in HR mode or in standard resolution mode by ignoring the HR field. By default, the hr_lr bit value is 0 which implies HR operation mode. However, setting this bit to one allows the use of several HR instructions on a single HR pin. Only one instruction is allowed to operate in HR mode (bit cleared to 0), but the other instructions can be used in standard resolution mode (bit set to 1).

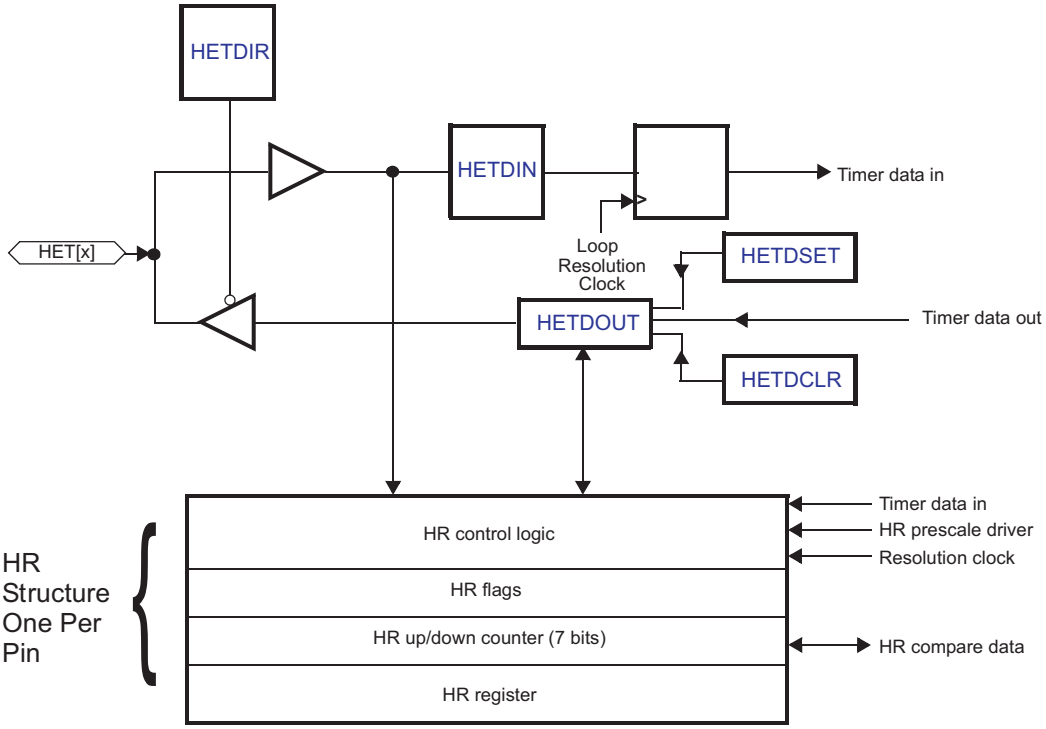
17.2.5.4 HR Block Diagram

Each time an HR instruction is executed on a given pin, the HR structure for that pin is programmed (which HR function to perform and on which edges it should take an action) with the information given by the instruction. The HR structure for each pin decodes the pin select field of the instruction and programs its HR structure if it matches.

NOTE: For each N2HET pin, only one instruction specifying a high resolution operation (hr_lr = HIGH) is allowed to execute per loop resolution period. This includes any instructions where (hr_lr = HIGH) but (en_pin_action = OFF).

The first high resolution instruction that executes and specifies a particular pin locks out subsequent high resolution instructions from operating on the same pin until the end fo the current loop resolution period.

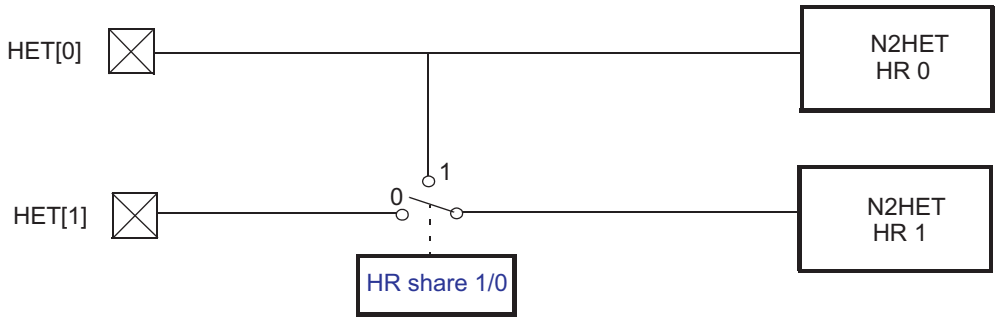
Figure 17-11. HR I/O Architecture



17.2.5.5 HR Structures Sharing (Input)

The HR Share Control Register (HETHRSH) allows two HR structures to share the same pin **for input capture only**. If these bits are set, the HR structures N and N+1 are connected to pin N. In this structure, pin N+1 remains available for general purpose input/output. See Figure 17-12.

Figure 17-12. Example of HR Structure Sharing for N2HET Pins 0/1



The following program gives an example how the HR share feature (HET[0] HR structure and HET[1] HR structure shared) can be used for the PCNT instruction:

```
L00 PCNT { next=L01, type=rise2fall, pin=0 }
L01 PCNT { next=L00, type=fall2rise, pin=1 }
```

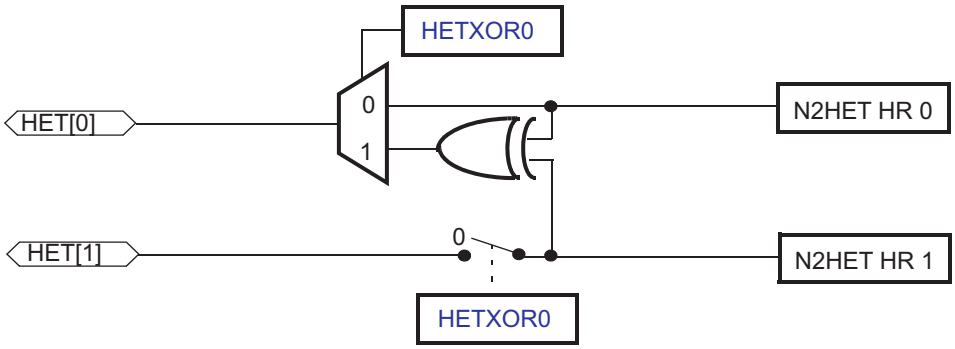
The HET[1] HR structure is also connected to the HET[0] pin. The L00_PCNT data field is able to capture a high pulse and the L01_PCNT captures a low pulse on the **same** pin (N2HET [0] pin).

17.2.5.6 AND / XOR-shared HR Structure (Output)

Usually the N2HET design allows only one HR structure to generate HR edges on a pin configured as output pin. The HETXOR register allows a logical XOR of the output signals of two consecutive HR structures N (even) and N+1 (odd). See Figure 17-13. In this way, it is possible to generate pulses smaller than the loop resolution clock since both edges can be generated by two independent HR structures. This is especially required for symmetrical PWM. See Figure 17-14.

The hardware provides a XOR gate that is connected to the outputs of the HR structure of two consecutive pins. In this structure, pin N+1 remains available for general purpose input/output.

Figure 17-13. XOR-shared HR I/O



The following N2HET program gives an example for **one** channel of the symmetrical PWM. The generated timing is given in Figure 17-14.

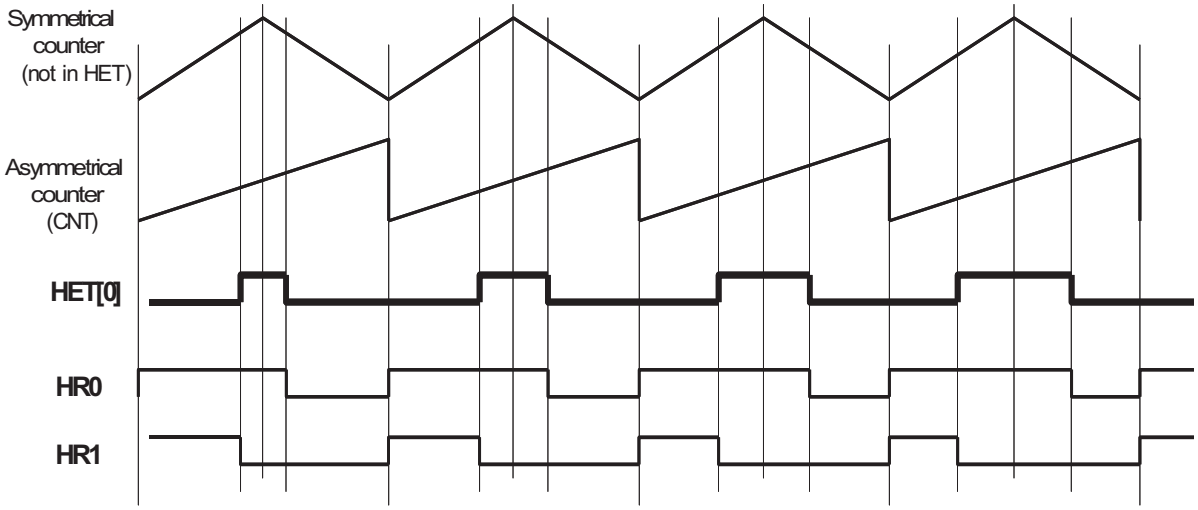
```
MAXC .equ 22
A_   .equ 0  ; HR structure HR0
B_   .equ 1  ; HR structure HR1

CN CNT   { next=EA, reg=A, max=MAXC }

EA ECMP  { next=EB, cond_addr=MA, hr_lr=HIGH, en_pin_action=ON, pin=A_,
           action=PULSELO, reg=A, data=17, hr_data=115 }

MA MOV32 { next=EB, remote=EA, type=IMTOREG&REM, reg=NONE, data=17, hr_data=19 }

EB ECMP  { next=CN, cond_addr=MB, hr_lr=HIGH, en_pin_action=ON, pin=B_,
           action=PULSELO, reg=A, data=5, hr_data=13 }

MB MOV32 { next=CN, remote=EB, type=IMTOREG&REM, reg=NONE, data=5, hr_data=13 }
```

N2HET Settings and output signal calculation for this example program:

- Pin HET[0] and HET[1] are XOR-shared.
- HETPFR[31:0] register = 0x700: lr = 128, hr = 1, time slots ts = 128
- PWM period (determined by CNT_max field) = (22+1) · LRP = 2944 HRP
- Length of high pulse of (HET[0] XOR HET[1]) =
  LH = (17·LRP + 115·HRP) - (5·LRP + 13·HRP)
  With lr = 128 there is LRP = 128 · HRP, so
  LH = (2291 - 653) · HRP = 1638 HRP
- Duty cycle = DC = LH / PWM_period = 1638 HRP / (2944·HRP) = 55.6 %

Figure 17-14 graphically shows the implementation of the XOR-shared feature. The first 2 waveforms (symmetrical counter and CNT) show a symmetric counter and asymmetric counter. The symmetric counter is shown only to highlight the axis of symmetry and is not implemented in the N2HET. The asymmetric counter, which is implemented with a CNT instruction, needs to be set to the period of the symmetric counter. The next two waveforms (HR [0] and HR [1]) show the output of the HR structures, which are the inputs for the XOR gate to create the PWM output on pin HET[0]. Notice that the pulses of signal HET[0] are centered about the axis of symmetry.

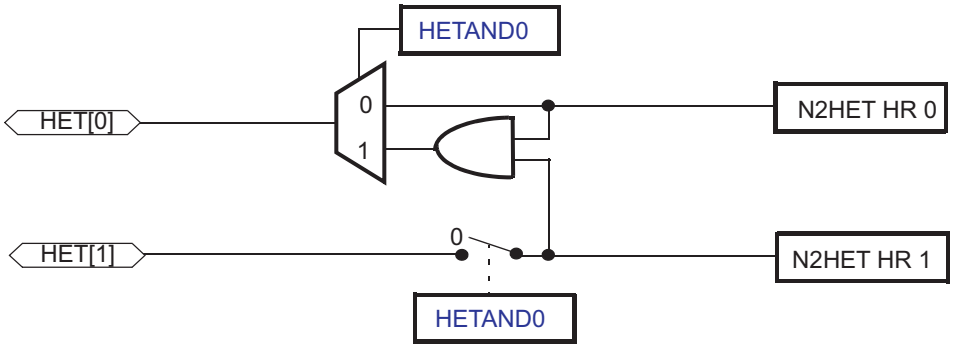
Figure 17-14. Symmetrical PWM with XOR-sharing Output



As an alternative, HR structures may be shared using a logical AND function to combine the effects of the pin structures. The HETAND allows sharing two consecutive HR structures N (even) and N+1 (odd). See Figure 17-15. In this structure, pin N+1 remains available for general purpose input/output.

NOTE: Setting both the HETAND bit and HETXOR bits at the same time for a given pair of N2HET pins is not supported, must be avoided by the application program.

Figure 17-15. AND-shared HR I/O



17.2.5.7 Loop Back Mode

The loop back feature can be used by the application to monitor an N2HET output signal. For example, if a PWM is generated by HR structure 0, then a PCNT instruction assigned to HR structure 1 can measure back the pulse length or periods of the PWM output signal.

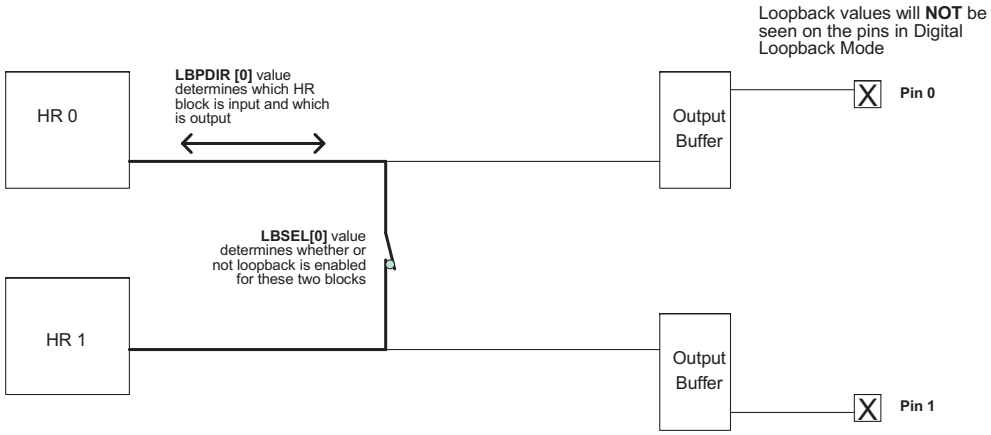
Loopback mode is activated between two high resolution structures by setting LBPSEL[x] to ‘1’ in the HETLBPSEL Register for the corresponding structure pair. The **direction** of the loopback between the two structures in the structure pair is determined by the value of LBPDIR[x] in the HETLBPDIR Register.

For example, if bit LBPSEL[0] is set to ‘1’, then HR structures 0 and 1 will be internally connected in loop back mode. If bit LBPDIR[0] is set to ‘0’ then structure 0 will be the input and structure 1 will be the output.

Digital Loopback

Digital loopback mode is enabled by setting LBPTYPE[x] to ‘0’ in the HETLBPSEL Register for the corresponding structure pairs. In digital loopback mode, the structure pairs are connected directly and the output buffers are bypassed. Therefore, the loopback values will NOT be seen on the corresponding pins. Figure 17-16 shows an example of digital loopback between structures HR0 and HR1. LBSEL[0] has been set to ‘1’ to enable loopback between the two structures. LBTYPE[0] has been set to ‘0’ to select digital mode for the loopback pair. The LPBDIR[0] value will determine the direction of the loopback by selecting which of the HR blocks is output, and which is input. The bold lines show the digital loopback path.

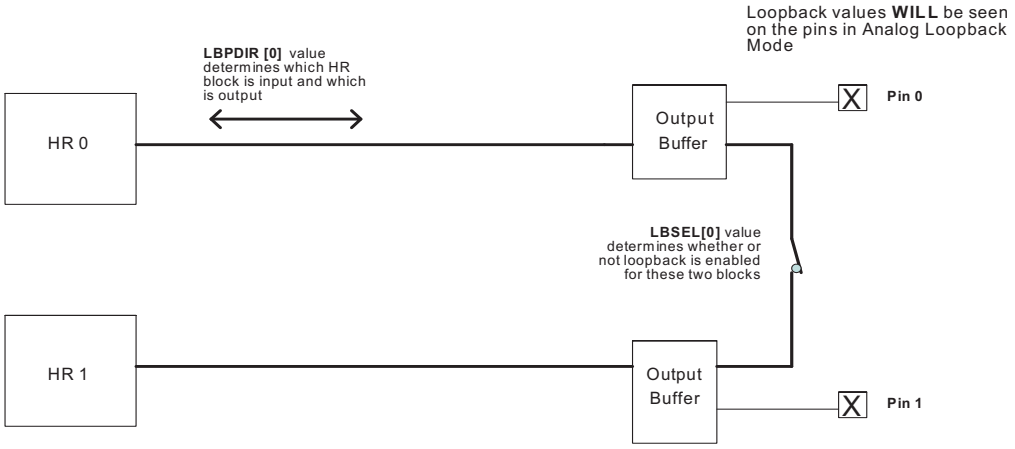
Figure 17-16. HR0 to HR1 Digital Loopback Logic: LBTYPE[0] = 0



Analog Loopback

Analog loopback mode is enabled by setting LBPTYPE[x] to ‘1’ in the HETLBPSEL Register for the corresponding structure pairs. In analog loopback mode, the structure pairs are connected outside of the output buffers. Therefore, the loopback values WILL be seen on the corresponding pins. Figure 17-17 shows an example of analog loopback between structures HR0 and HR1. LBSEL[0] has been set to ‘1’ to enable loopback between the two structures. LBTYPE[0] has been set to ‘1’ to select analog mode for the loopback pair. The LPBDIR[0] value will determine the direction of the loopback by selecting which of the HR blocks is output, and which is input. The bold lines show the analog loopback path.

Figure 17-17. HR0 to HR1 Analog Loop Back Logic: LBTYPE[0] = 1



Note:

- The loop back direction can be selected independent of the HETDIR register setting.
- The pin which is not driven by the N2HET output pin actions can still be used as normal GIO pin.

17.2.5.8 Edge Detection Input Timing

There are several timing requirements for input signals in order to be captured correctly by N2HET. Figure 17-18 illustrates these requirements, with min and max values described in Table 17-7 (Loop Resolution) and Table 17-8 (High Resolution).

Figure 17-18. N2HET Input Edge Detection

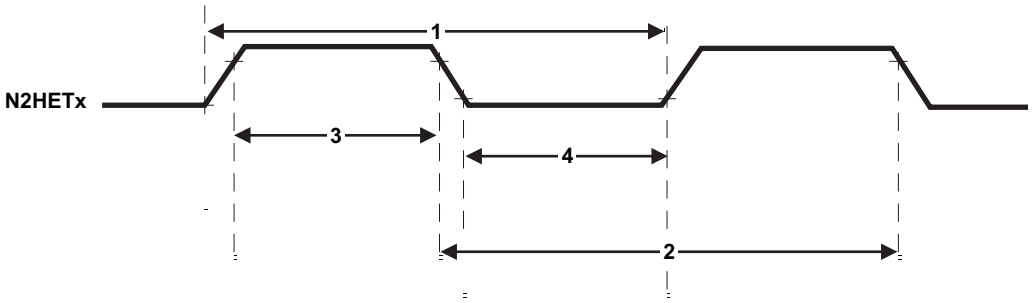


Table 17-7. Edge Detection Input Timing for Loop Resolution Instructions

Parameter #	Description	min	max
1	Input Signal Period, rising edge to rising edge	> 2 (hr) (lr) $t_{c(VCLK2)}$	< 2^{25} (hr) (lr) $t_{c(VCLK2)}$
2	Input Signal Period, falling edge to falling edge		
3	Input Signal, high phase	> (hr) (lr) $t_{c(VCLK2)}$	
4	Input Signal, high phase		

Table 17-8. Edge Detection Input Timing for High Resolution Instructions

Parameter #	Description	min	max
1	Input Signal Period, rising edge to rising edge	> (hr) (lr) $t_{c(VCLK2)}$	< 2^{25} (hr) (lr) $t_{c(VCLK2)}$
2	Input Signal Period, falling edge to falling edge		
3	Input Signal, high phase	> 2 (hr) $t_{c(VCLK2)}$	
4	Input Signal, high phase		

These are the N2HET architectural limitations. Actual limitations will be slightly different due to on chip routing and IO buffer delays, usually by several nanoseconds. Be sure to consult the device datasheet for actual timings that apply to that device. Also, certain devices place additional restrictions on which pins support the high resolution timings of Table 17-8, if present these additional limitations will also be called out in the device datasheet.

Note that the max limit in Table 17-7 and Table 17-8 is based on the counter range of a single N2HET instruction. The max value could be extended by employing an additional N2HET instruction to keep track of counter overflows of the input counter / capture instruction.

17.2.5.9 PWM Generation Example 1 (in HR Mode)

The following example shows how an ECMP instruction works in high resolution mode. The example assumes a VCLK2 of 32 MHz and the following values for the prescale divide rates (hr and lr), number of time slots (ts), high and loop resolution period (HRP and LRP):

hr = 2, lr = 4, ts = hr × lr = 8

HRP = hr / VCLK2 = 2 / 32 MHz = 62.5 ns

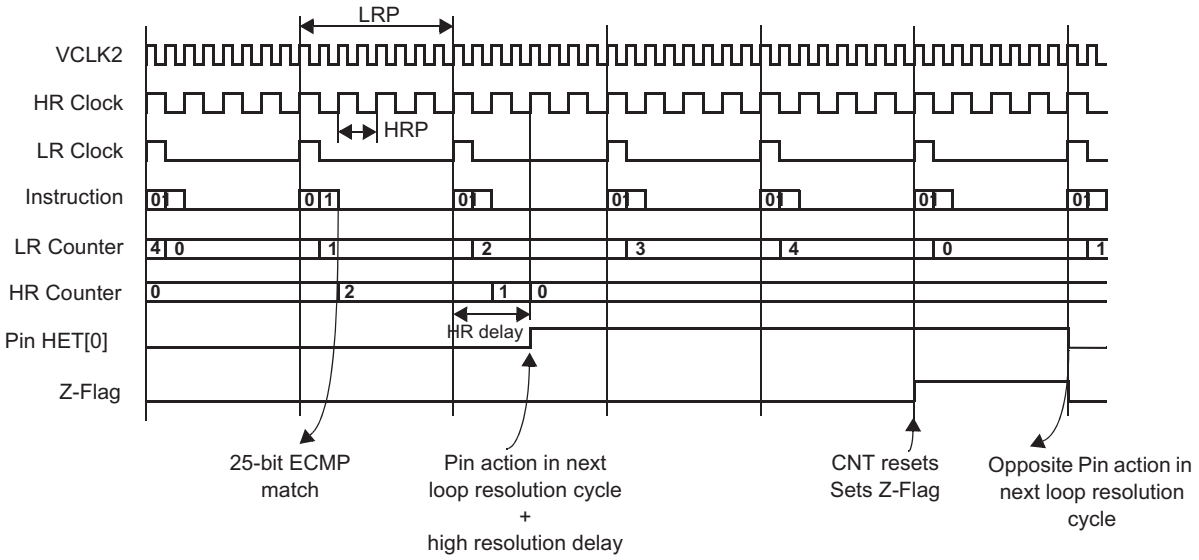
LRP = (hr × lr) / VCLK2 = 8 / 32 MHz = 250 ns

With ts = 8 there are eight time slots available for the program execution, which in this case will consist of one CNT and one ECMP instruction as shown below. The data field of the ECMP instruction is the 32-bit compare value, whereby the lower 7 bits represent the high resolution compare field.

When the 25-bit (loop resolution) compare matches, the HR compare value will be loaded from the 7 lower bits of the instruction data field to the HR counter. At the next loop resolution clock, the HR counter will count down at the HR clock frequency and perform the pin action when it reaches zero.

In the example illustrated by Figure 17-19, the 25-bit compare value is one and the 7-bit HR compare value is two. According to Section 17.2.3.2, depending on the loop resolution divide rate (lr), only certain bits of the 7-bit HR compare value are valid. In this example only the upper 2 bits (D[6:5]) are taken into account. The example program below has a setting of hr_data = 100000b. Shifting this value right by 5 bits, results in 10b which equals the two HR clock cycles delay mentioned above.

Figure 17-19. ECMP Execution Timings




```
HETPFR[31:0] register = 0x201 --> lr=4 and hr=2 --> ts = 8

N2HET Program:

L00   CNT   { next= L01, reg=A, irq=OFF, max = 4 }
L01   ECMP  { next= L00, cond_addr= L00, hr_lr=HIGH, en_pin_action=ON, pin=0,
              action=PULSEHI, reg=A, irq=OFF, data= 1, hr_data = 0x40 }

; 25 bit compare value is 1 and the 7-bit HR compare value is 2
; (Because of lr=4 the D[4:0] of the 7-bit HR field are ignored )
```

NOTE: ECMP Opposite Actions

ECMP opposite pin actions are always synchronized to the loop resolution clock.

Changing the duty cycle of a PWM generated by an ECMP instruction, can lead to a missing pulse if the data field of the instruction is updated directly. This can happen when it is changed from a high value to a lower value while the CNT instruction has already passed the new updated lower value. To avoid this a synchronous duty cycle update can be performed with the use of an additional instruction (MOV32). This instruction is only executed when the compare of the ECMP matches. For this the cond_addr of the ECMP needs to point to the MOV32. On execution of the MOV32, it moves its data field into the data field of the ECMP. The update of the duty cycle has to be made to the MOV32 data field instead of the ECMP data field.

17.2.5.10 PWM Generation Example 2 (in HR Mode)

The MCMP instruction can also be used in HR mode. In this case operation is exactly the same as for the ECMP instruction except that the 25-bit low resolution is now the result of a magnitude compare (greater or equal) rather than an equality compare. When the 25-bit (loop resolution) magnitude compare matches, the HR compare value will be loaded from the 7 lower bits of the instruction data field to the HR counter. At the next loop resolution clock, the HR counter will count down at the HR clock frequency and perform the pin action when it reaches zero.

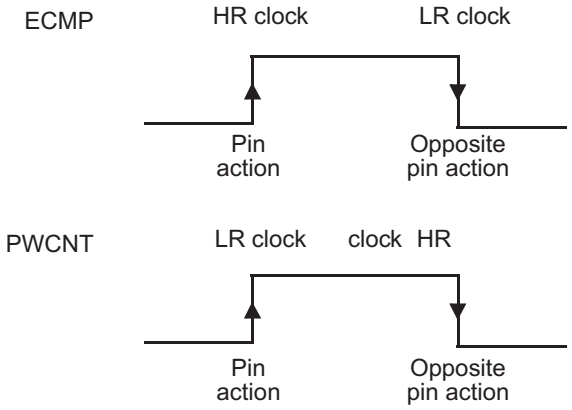
The MCMP instruction avoids the missing pulse problem of the ECMP instruction (see previous example), however the duty cycle of the signal might not be exact for one PWM period. The benefit of the MCMP is that it avoids adding another instruction to do the duty cycle update synchronously.

17.2.5.11 Pulse Generation Example (in HR Mode)

The PWCNT instruction may also be used in HR mode to generate pulse outputs with HR width. It generates a single pulse when the data field of the instruction is non-zero. It remains at the opposite pin action when the data field is zero.

The PWCNT instruction operates conversely to the ECMP instruction. See Figure 17-20. For PWCNT, the opposite pin action is synchronous with the HR clock and for ECMP the pin action is synchronous with the HR clock. The PWCNT pin action is synchronous with the loop resolution clock.

Figure 17-20. High/Low Resolution Modes for ECMP and PWCNT



17.2.5.12 Pulse Measurement Example (in HR Mode)

The PCNT instruction captures HR measurement of the high/low pulse time or periods of the input. As shown in Figure 17-21, at marker (1) the input goes HIGH and the HR counter immediately begins to count. The counter increments and rolls over until the falling edge on the input pin, where it captures the counter value into the HR capture register (marker (2)). The PCNT instruction begins counting when the synchronized input signal goes HIGH and captures both the 25-bit data field and the HR capture register into RAM when the synchronized input falls (marker (3)).

NOTE: The HR capture value written into RAM is shifted appropriately depending on the loop resolution prescale divide rate (lr). (See also Section 17.2.3.2.)

Figure 17-21 shows what happens when the capture edge arrives *after* the HR counter overflows. This causes the incremented value to be captured by the PCNT instruction.

Figure 17-21. PCNT Instruction Timing (With Capture Edge After HR Counter Overflow)

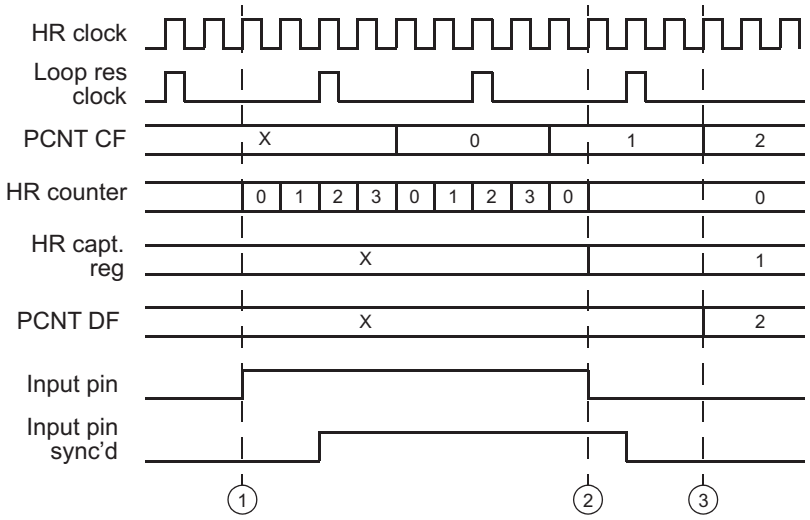
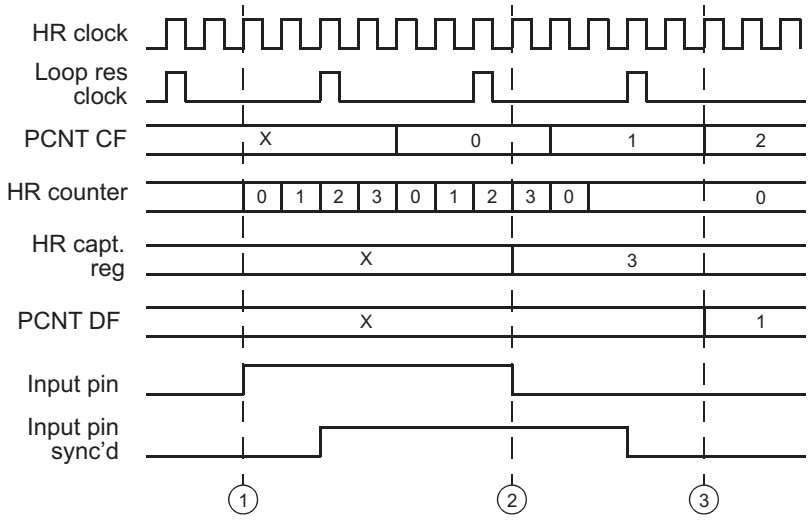


Figure 17-22 shows what happens when the capture edge arrives *before* the HR counter overflows. This causes the non-incremented value to be captured by the PCNT instruction.

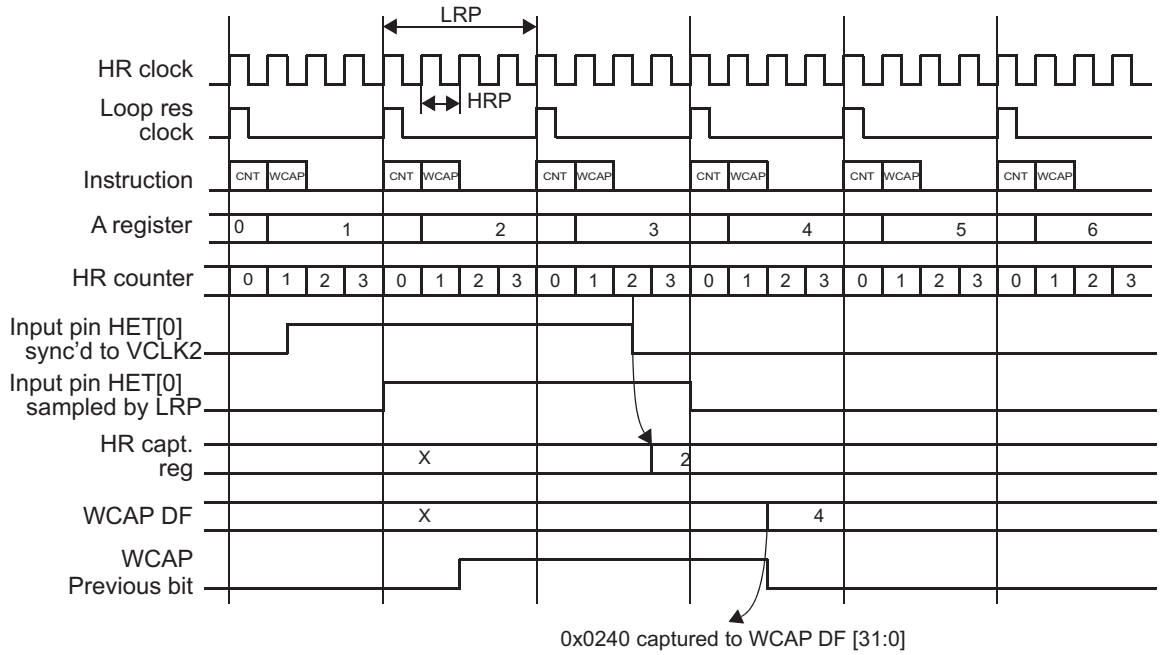
Figure 17-22. PCNT Instruction Timing (With Capture Edge Before HR Counter Overflow)



17.2.5.13 WCAP Execution Example (in HR Mode)

The HR capability is enabled for WCAP, if its hr_lr bit is zero. In this case the HR counter is always enabled and is synchronized with the resolution loop. When the specified edge is detected, the current value of the HR counter is captured in the HR capture register and written into the RAM after the next WCAP execution. The WCAP instruction effectively time stamps the free running timer saved in a register (for example, register A shown in Figure 17-23).

Figure 17-23. WCAP Instruction Timing

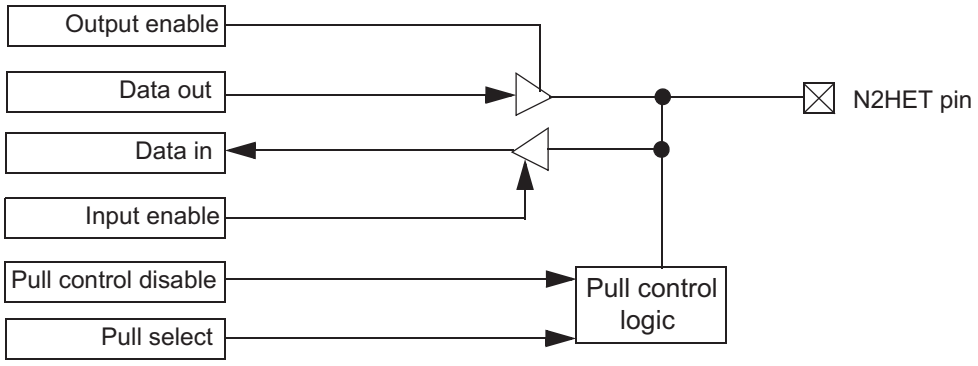



```
HETPFR_register = 0x0200 --> lr = 4, hr = 1, ts = 4

N2HET Program:

L00 CNT  {reg=A, max=01ffffffh}
L01 WCAP {next=L00, cond_addr=L00, hr_lr=high, reg=A, event= FALL, pin=0,
          data=0}
```

In the example, the WCAP is configured to capture the counter when a **falling** edge occurs. The WCAP data field (WCAP_DF) is updated in the loop succeeding the loop in which the edge occurred. The WCAP instruction evaluates an edge by comparing its Previous bit with the sync'd input signal. In Figure 17-23, the current value of the counter (4) is captured to WCAP_DF[31:7] and the value of the HR capture register (2) is transferred to the valid bits (according the lr prescaler) of WCAP_DF[6:0]. Therefore, in the example 0x0240 is captured in WCAP_DF[31:0].

17.2.5.14 I/O Pull Control Feature

Figure 17-24. I/O Block Diagram Including Pull Control Logic



The following apply if the device is under reset:

- Pull control: The reset pull control on the pins is enabled and a pulldown is configured.
- Input buffer: The input buffer is enabled.
- Output buffer: The output buffer is disabled.

The following apply if the device is out of reset:

- Pull control: The pull control is enabled by clearing the corresponding bit in the N2HET Pull Disable Register (HETPULDIS). In this case, if the corresponding bit in the N2HET Pull Select Register (HETPSL) is set, the pin will have a pull-up; if the bit in the N2HET Pull Select Register (HETPSL) is cleared, the pin will have a pull-down. If the bit in the N2HET Pull Disable Register (HETPULDIS) is set, there is no pull-up or pull-down on the pin.
- Input buffer: In all cases, the input buffer is enabled.

NOTE: The pull-disable logic depends on the pin direction. If the pin is configured as output, then the pulls are disabled automatically. If the pin is configured as input, the pulls are enabled or disabled depending on the pull disable register bit.

- Output buffer: A pin can be driven as an output pin if the corresponding bit in the N2HET Direction Register (HETDIR) is set AND the open-drain feature (N2HET Open Drain Register (HETPDR)) is not enabled. See Section 17.2.5.15 for more details.

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 17-9.

Table 17-9. Input Buffer, Output Buffer, and Pull Control Behavior

Device under Reset?	Pin Direction (DIR)[1]	Pull Disable (PULDIS)[1]	Pull Select (PULSEL)[1]	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Enabled	Disabled	Enabled
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Disabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

(1) X = Don't care

17.2.5.15 Open-Drain Feature

The following apply if the open-drain feature is enabled on a pin, that is, the corresponding bit in the N2HET Open Drain Register (HETPDR) is set:

- Output buffer is enabled, if a low signal is being driven internally to the pin.
- Output buffer is disabled, if a high signal is being driven internally to the pin.

17.2.5.16 N2HET Pin Disable Feature

This feature is provided for the safe operation of systems such as power converters and motor drives. It can be used to inform the monitoring software of motor drive abnormalities such as over-voltage, over-current, and excessive temperature rise.

Figure 17-25. N2HET Pin Disable Feature Diagram

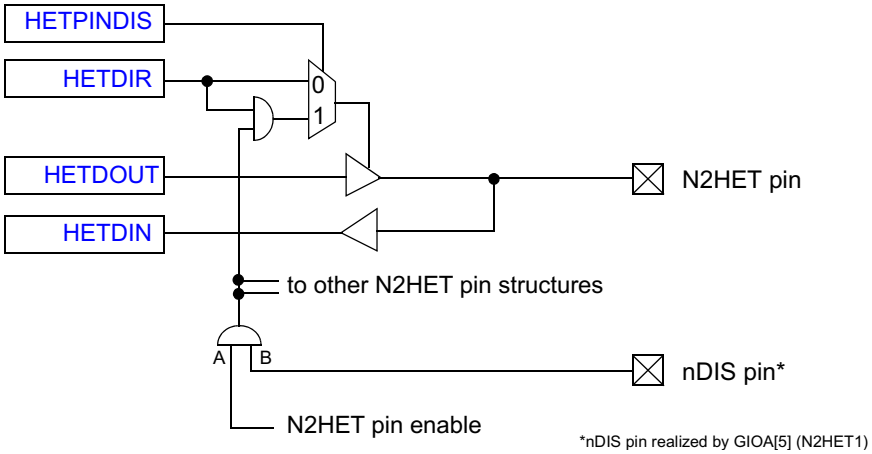


Table 17-10 shows the conditions for the output buffer to be enabled/disabled.

Table 17-10. N2HET Pin Disable Feature

HETPINDIS.x	nDIS Pin (Input)	HET_PIN_ENA (HETGCR.24)	HETDIR.x	Output Buffer
0	X	X	0	Disabled
0	X	X	1	Enabled
1	0	X	0	Disabled
1	0	X	1	Disabled
1	1	X	0	Disabled
1	1	0	1	Disabled
1	1	1	1	Enabled

An interrupt capable device I/O pin can share the same pin as the N2HET nDIS signal. Normally GIOA[5] serves as nDIS. Check the device datasheet for the actual implementation. Sharing a pin with a GIO pin that is Interrupt capable allows the N2HET nDIS input to also generate an interrupt to the CPU. An active low level on nDIS is intended to signal an abnormal situation as described above. All N2HET pins, which are selected with the N2HET Pin Disable Register (HETPINDIS), will be put in the high-impedance state by hardware immediately after the nDIS signal is pulled low. At this time a CPU interrupt is issued, if it is enabled in the GIO pin logic.

The bit HET_PIN_ENA is automatically cleared in the failure condition and this state remains as long as the software explicitly sets the bit again. The steps to do this are:

- Software detects, by reading the HETDIN register of the GIO pin, that the level on nDIS is inactive (high).
- Software sets bit HET_PIN_ENA to deactivate the high impedance state of the pins.

17.2.6 Suppression Filters

Each N2HET pin is equipped with a suppression filter. If the pin is configured as an input it enables to filter out pulses shorter than a programmable duration. Each filter consists of a 10-bit down counter, which starts counting at a programmable preloaded value and is decremented using the VCLK2 clock.

- The counter starts counting when the filter input signal has the opposite state of the filter output signal. The output signal is preset to the same input signal state after reset, in order to ensure proper operation after device reset.
- Once the counter reaches zero without detecting an opposite pin state on the filter input signal, the output signal is set to the opposite state.
- When the counter detects an opposite pin action on the filter input signal before reaching zero, the counter is loaded with it's preload value and the opposite pin action on the filter output signal does not take place. The counter resumes at the preload value until it detects an opposite pin action on the input signal again.
- Therefore the filter output signal is delayed compared to the filter input signal. The amount of delay depends on the counter clock frequency (VCLK2) and the programmed preload value.
- The accuracy of the output signal is +/- the counter clock frequency.

Figure 17-26. Suppression Filter Counter Operation

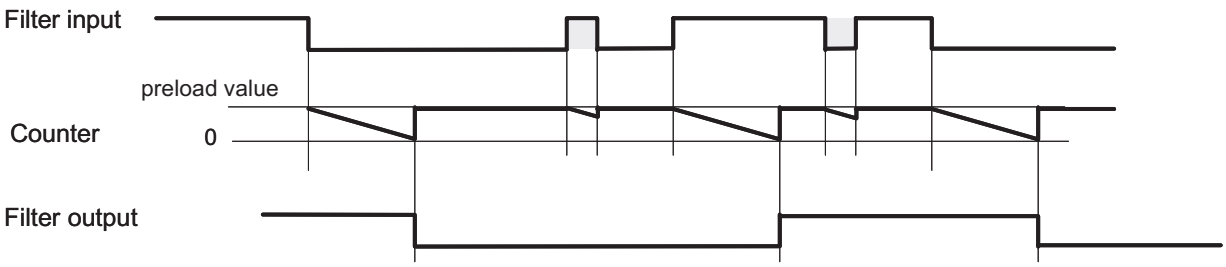


Table 17-11 gives examples for a 100 MHz VCLK2 frequency.

Table 17-11. Pulse Length Examples for Suppression Filter

Divider CCDIV	VCLK2	Possible values for the suppressed pulse length / frequency resulting from the programmable 10 bit preload value (0,1,..,1023)	
1	100.0 MHz	10 ns, 20 ns, …, 10.22 µs, 10.23 µs	50 MHz, 25 MHz, …, 48.924 kHz, 48.876 kHz
2	50.0 MHz	20 ns, 40 ns, …, 20.44 µs, 20.48 µs	25 MHz, 12.5 MHz, …, 24.462 kHz, 24.414 kHz
3	33.3 MHz	30 ns, 60 ns, …, 30.66 µs, 30.69 µs	16.7 MHz, 8.3 MHz, …, 16.308 kHz, 16.292 kHz

17.2.7 Interrupts and Exceptions

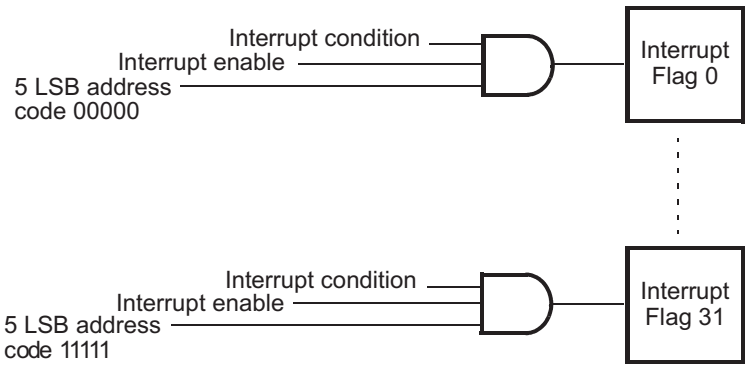
N2HET interrupts can be generated by any instruction that has an interrupt enable bit in its instruction format. When the interrupt condition in an instruction is true and the interrupt enable bit of that instruction is set an interrupt flag is then set in the N2HET Interrupt Flag Register (HETFLG). The address code for this flag is determined by the five LSBs of the current timer program address. The flag in the N2HET Interrupt Flag Register (HETFLG) is set even if the corresponding bit in the N2HET Interrupt Enable Set Register (HETINTENAS) is zero. To generate an interrupt the corresponding bit in the N2HET Interrupt Enable Set Register (HETINTENAS) must be one. In the N2HET interrupt service routine, the main CPU must first determine which source inside the N2HET created the interrupt request. This operation is accelerated by the N2HET Offset Index Priority Level 1 Register (HETOFF1) or N2HET Offset Index Priority Level 2 Register (HETOFF2) which automatically provide the number of the highest priority source within each priority level. Reading the offset register will automatically clear the corresponding N2HET interrupt flag which created the request. However, if the offset registers are not used by the N2HET interrupt service routine, the flag should be cleared explicitly by the CPU once the interrupt has been serviced.

Table 17-12. Interrupt Sources and Corresponding Offset Values in Registers HETOFFx

Source No.	Offset Value
no interrupt	0
Instruction 0, 32, 64...	1
Instruction 1, 33, 65...	2
:	:
Instruction 31, 63, 95...	32
Program Overflow	33
APCNT underflow:	34
APCNT overflow	35

The instructions capable of generating interrupts are listed in Table 17-50.

Figure 17-27. Interrupt Functionality on Instruction Level



Each interrupt source is associated with a priority level (level 1 or level 2). When multiple interrupts with the same priority level occur during the same loop resolution the lowest flag bit is serviced first.

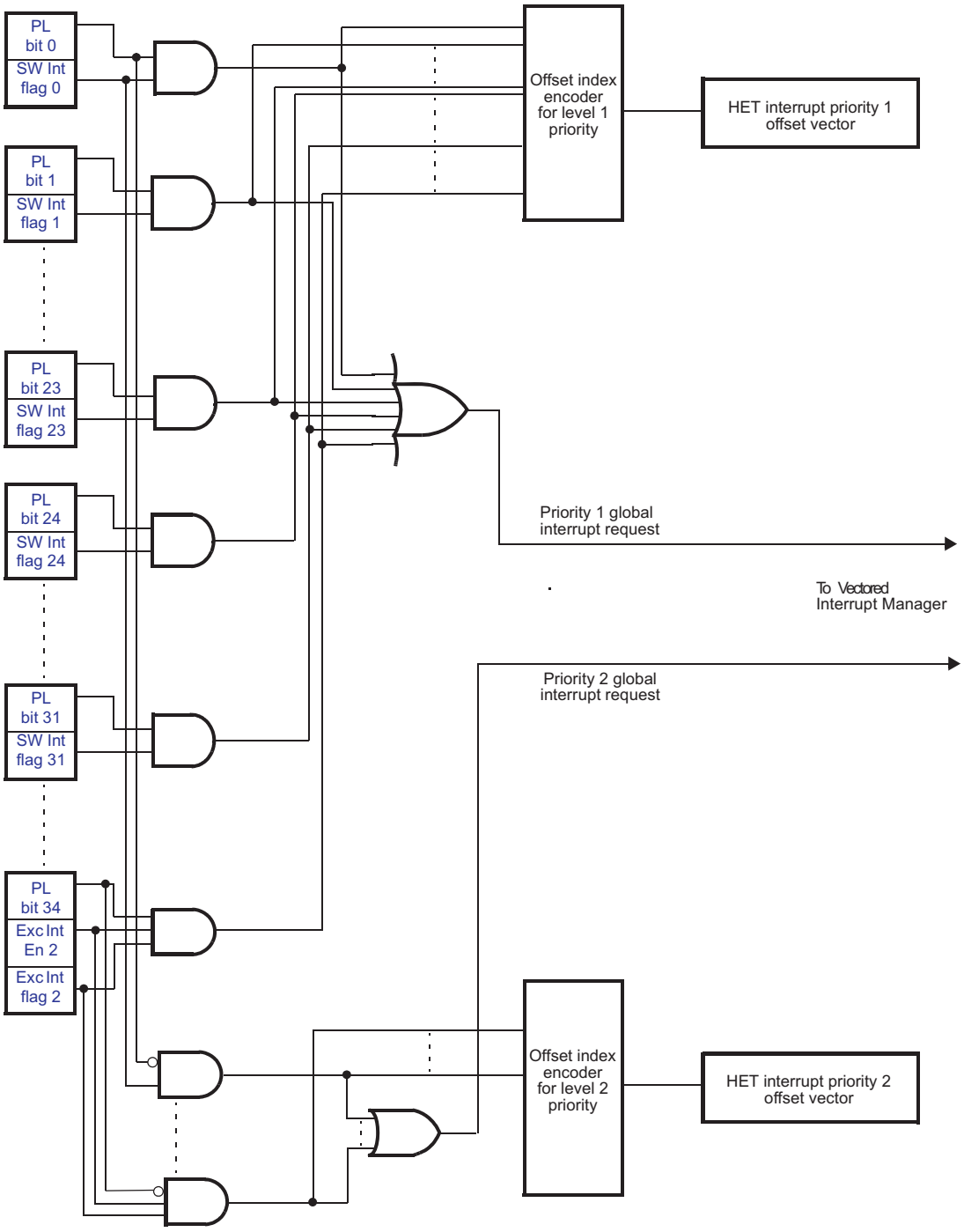
In addition to the interrupts generated by the instructions, the N2HET can generate three additional exceptions:

- Program overflow
- APCNT underflow (see Section 17.3.1.2)
- APCNT overflow (see Section 17.3.1.3)

17.2.8 Hardware Priority Scheme:

If two or more software interrupts are pending on the same priority level, the offset value will show the one with the highest priority. The interrupt with the highest priority is the one with the lower offset value. This scheme is hard-wired in the offset encoder. See Figure 17-28.

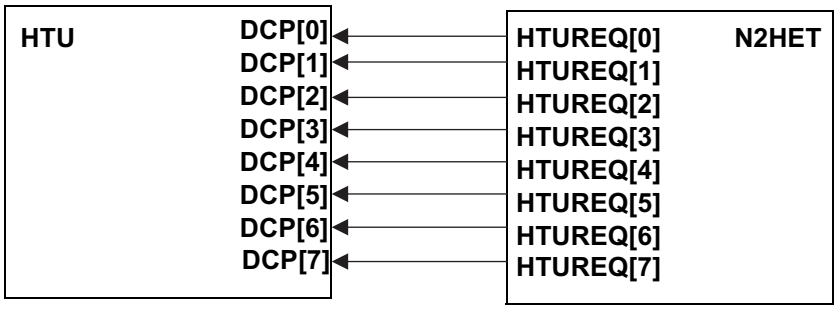
Figure 17-28. Interrupt Flag/Priority Level Architecture



17.2.9 N2HET Requests to the HTU

As described in Section 17.5.3, the majority of the N2HET instructions are able to generate a transfer request to the High-End Timer Transfer Unit (HTU) when an instruction-specific condition is true. One N2HET instruction can select one of 8 request lines by programming the “reqnum” parameter. The “request” field in an instruction is used to enable, disable, or to generate a quiet request (see Section 17.5.2) on the selected request line provided the request line is enabled using the HETREQENS register. A request line can be disabled using the HETREQENC register. Quiet requests can be used by the HTU. For quiet request, refer to Section 18.2.4.1.

Figure 17-29. Request Line Assignment Example



17.3 Angle Functions

Engine management systems require an angle-referenced time base to synchronize signals to the engine toothed wheel. The N2HET has a method to provide such a time base for low-end engine systems. The reference is created by the N2HET using three dedicated instructions with fractional angle steps equal to /8, /16, /32, /64.

17.3.1 Software Angle Generator

The N2HET provides three specialized count instructions to generate an angle referenced time base synchronized to an external reference signal (the toothed wheel signal) that defines angular reference points.

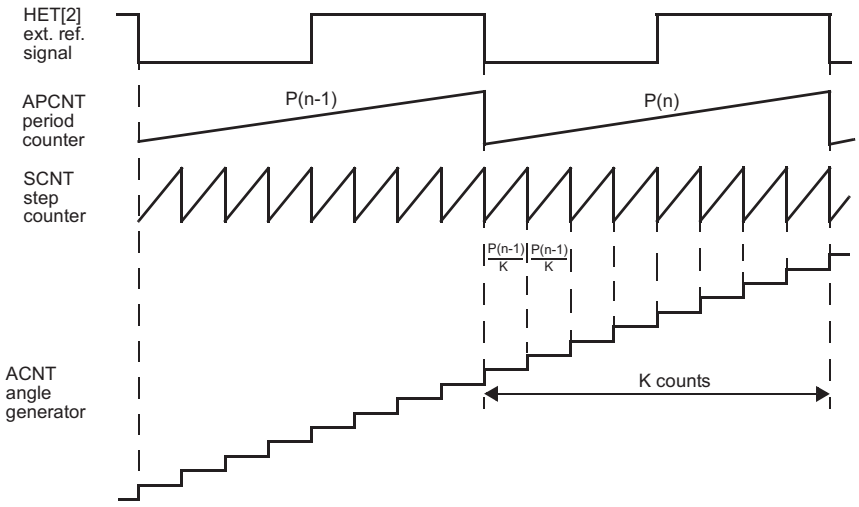
The time base is used to generate fractional angle steps between the reference points. The step width K (= 8, 16, 32, or 64) programmed by the user defines the angle accuracy of the time base. These fractional steps are then accumulated in an angle counter to form the absolute angle value.

The first counter, APCNT, incremented on each loop resolution clock measures the periods P(n) of the external signal. The second counter SCNT counts by step K up to the previous period value P(n-1), measured by APCNT, and then recycles. The resulting period of SCNT is the fraction P(n-1) / K. The third counter ACNT accumulates the fractions generated by SCNT.

Figure 17-30 illustrates the basic operation of APCNT, SCNT, and ACNT.

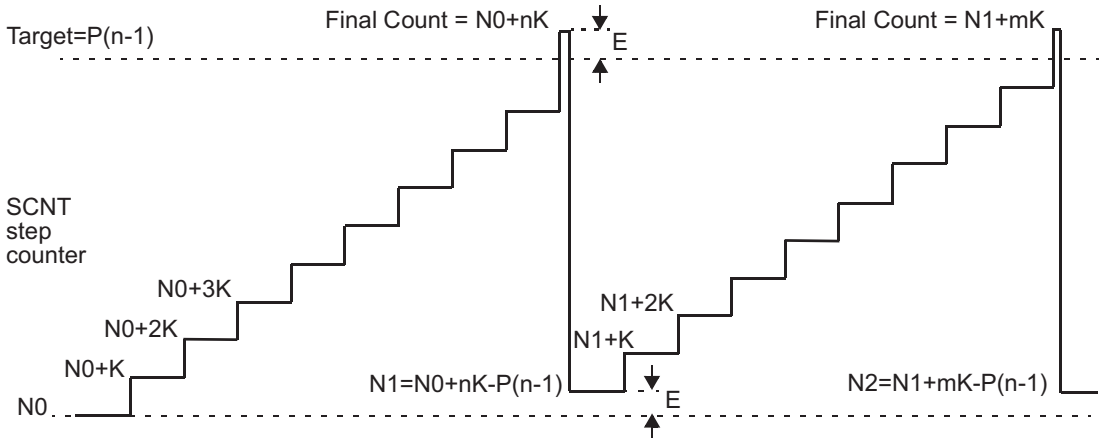
A N2HET timer program can only have one angle generator.

Figure 17-30. Operation of N2HET Count Instructions



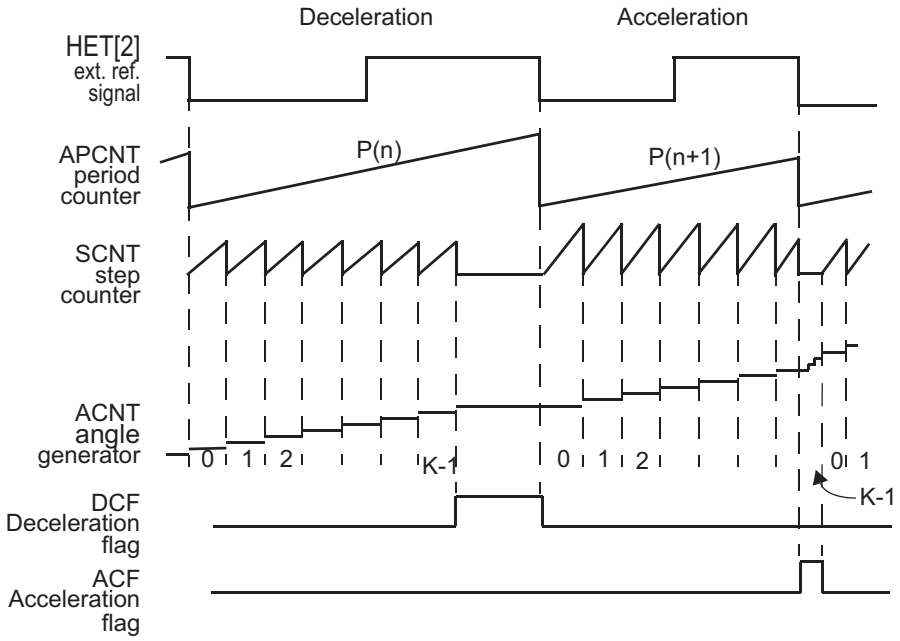
Due to stepping, the final count of SCNT does not usually exactly match the target value P(n-1). Figure 17-31 illustrates how SCNT compensates for this feature by starting each cycle with the remainder (final count - target) of the previous cycle.

Figure 17-31. SCNT Count Operation



ACNT detects period variations of the external signal measured by APCNT and compensates related counting errors. A period increase is flagged in the deceleration flag. A period decrease is flagged in the acceleration flag. If no variation is flagged, ACNT increments the counter value each time SCNT reaches its target. If acceleration is detected, ACNT increments the counter value on each timer resolution (fast mode). If deceleration is detected, ACNT is stopped. Figure 17-32 illustrates how the compensations for acceleration and deceleration operate.

Figure 17-32. ACNT Period Variation Compensations



17.3.1.1 Singularities

Singularities (gaps, in this case, from missing teeth in a toothed wheel) in the external reference signal can be masked. The start and end of singularities are defined by gap start and gap end values specified in SCNT and ACNT. When ACNT reaches gap start or gap end, it sets/resets the gap flag.

While the gap flag is set, new periods of the external reference signal are ignored for angle computation. SCNT uses the last period measured by APCNT just before gap start.

Figure 17-33 and Figure 17-34 illustrate the behavior of the angle generator during a gap after a deceleration or acceleration of the N2HET.

17.3.1.2 APCNT Underflow

The fastest valid external signal APCNT can accept must satisfy the following condition:

Step Width K < Period Min. Resolution (LRP)

This condition fixes the maximum possible step width once the minimum period and the resolution of an application are specified.

If a period value accidentally falls below the minimum allowed, APCNT stops the capture of these periods and sets the APCNT underflow interrupt flag located in the exceptions interrupt control register. In such a situation, SCNT and ACNT continue to be executed using the last valid period captured by APCNT.

17.3.1.3 APCNT Overflow

The slowest valid external signal APCNT can measure must satisfy the following condition:

Period Max Resolution < 33554431

When this limit is reached (APCNT Count equals all 1's), APCNT stays at a maximum count (stops counting). APCNT remains in this position until the next specified capture edge is detected on the selected pin and sets the APCNT overflow interrupt flag located in the exceptions interrupt control register. In this situation, SCNT and ACNT continue to be executed using the maximum APCNT period count.

Figure 17-33. N2HET Timings Associated with the Gap Flag (ACNT Deceleration)

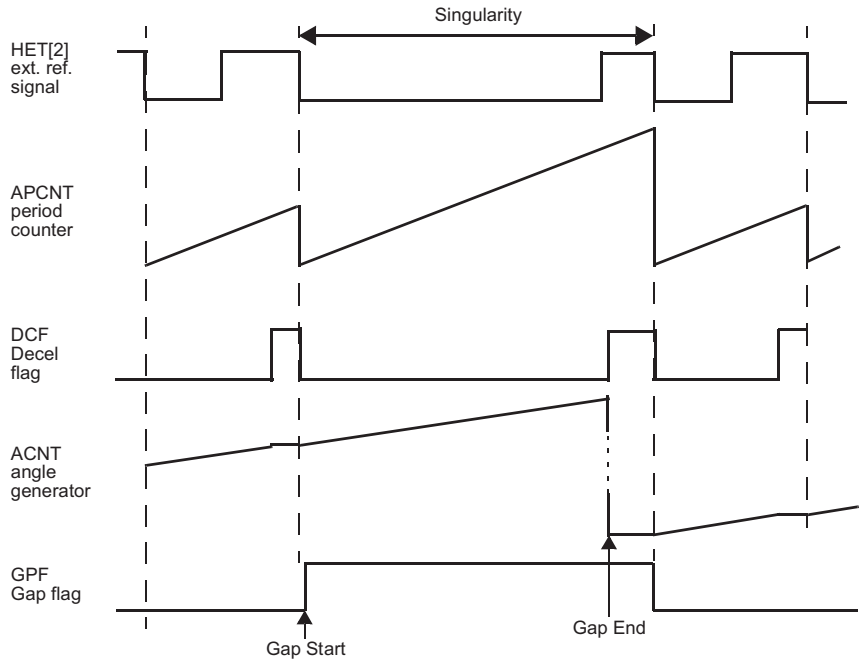
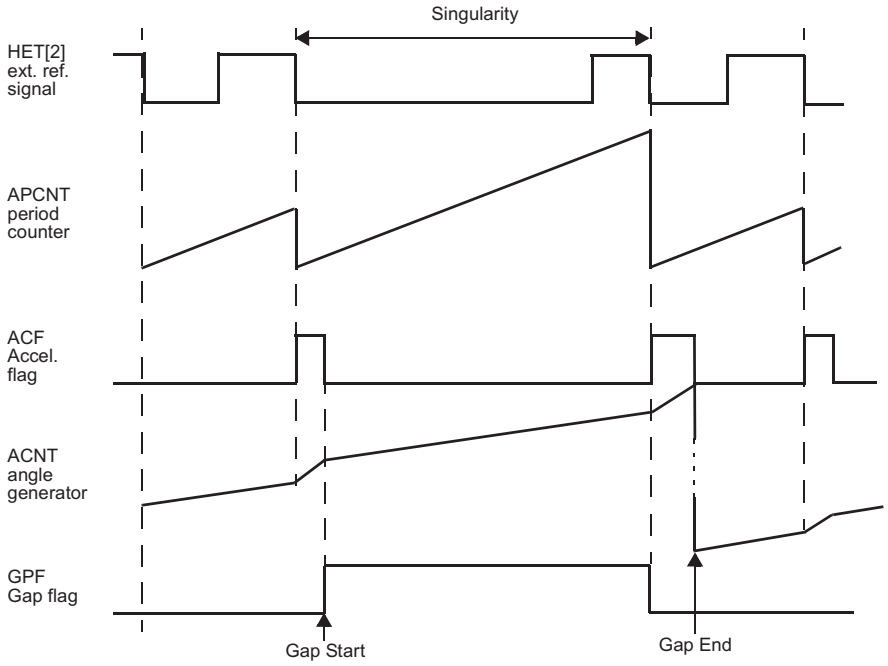


Figure 17-34. N2HET Timings Associated with the Gap Flag (ACNT Acceleration)



17.4 N2HET Control Registers

Table 17-13 summarizes all the N2HET registers. The base address for the control registers is FFF7 B800h.

Table 17-13. N2HET Registers

Offset	Acronym	Register Description	Section
00h	HETGCR	Global Configuration Register	Section 17.4.1
04h	HETPFR	Prescale Factor Register	Section 17.4.2
08h	HETADDR	NHET Current Address Register	Section 17.4.3
0Ch	HETOFF1	Offset Index Priority Level 1 Register	Section 17.4.4
10h	HETOFF2	Offset Index Priority Level 2 Register	Section 17.4.5
14h	HETINTENAS	Interrupt Enable Set Register	Section 17.4.6
18h	HETINTENAC	Interrupt Enable Clear Register	Section 17.4.7
1Ch	HETEXC1	Exception Control Register 1	Section 17.4.8
20h	HETEXC2	Exception Control Register 2	Section 17.4.9
24h	HETPRY	Interrupt Priority Register	Section 17.4.10
28h	HETFLG	Interrupt Flag Register	Section 17.4.11
2Ch	HETAND	AND Share Control Register	Section 17.4.12
34h	HETHRSH	HR Share Control Register	Section 17.4.13
38h	HETXOR	HR XOR-Share Control Register	Section 17.4.14
3Ch	HETREQENS	Request Enable Set Register	Section 17.4.15
40h	HETREQENC	Request Enable Clear Register	Section 17.4.16
44h	HETREQDS	Reserved. Do not use.	
4Ch	HETDIR	NHET Direction Register	Section 17.4.17
50h	HETDIN	NHET Data Input Register	Section 17.4.18
54h	HETDOUT	NHET Data Output Register	Section 17.4.19
58h	HETDSET	NHET Data Set Register	Section 17.4.20
5Ch	HETDCLR	NHET Data Clear Register	Section 17.4.21
60h	HETPDR	NHET Open Drain Register	Section 17.4.22
64h	HETPULDIS	NHET Pull Disable Register	Section 17.4.23
68h	HETPSL	NHET Pull Select Register	Section 17.4.24
74h	HETPCR	Parity Control Register	Section 17.4.25
78h	HETPAR	Parity Address Register	Section 17.4.26
7Ch	HETPPR	Parity Pin Register	Section 17.4.27
80h	HETSFPRLD	Suppression Filter Preload Register	Section 17.4.28
84h	HETSFENA	Suppression Filter Enable Register	Section 17.4.29
8Ch	HETLBPSEL	Loop Back Pair Select Register	Section 17.4.30
90h	HETLBPDIR	Loop Back Pair Direction Register	Section 17.4.31
94h	HETPINDIS	NHET Pin Disable Register	Section 17.4.32

17.4.1 Global Configuration Register (HETGCR)

Figure 17-35. Global Configuration Register (HETGCR) [offset = 00h]

31						25	24
Reserved							HET_PIN_ENA
R-0							R/W-1
23	22	21	20	19	18	17	16
Reserved	MP		Reserved		PPF	IS	CMS
R-0	R/W-0		R-0		R/W-0	R/W-0	R/W-0
15						1	0
Reserved							TO
R-0							R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-14. Global Configuration Register (HETGCR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	HET_PIN_ENA		Enables the output buffers of the pin structures depending on the value of nDIS and DIR.x when PINDIS.x is set.
			Note: This bit will automatically get cleared when nDIS pin (input port) value is 0.
		0	No affect on the pin output buffer structure.
		1	Enables the pin output buffer structure when DIR = output, PINDIS.x is set and nDIS = 1.
23	Reserved	0	Reads return 0. Writes have no effect.
22-21	MP		Master Priority
			The NHET can prioritize master accesses to N2HET RAM between the HET Transfer Unit and another arbiter, which outputs the access of one of the remaining masters. The MP bits allow the following selections:
		0	The HTU has lower priority to access the N2HET RAM than the arbiter output.
		1h	The HTU has higher priority to access the N2HET RAM than the arbiter output.
		2h	The HTU and the arbiter output use a round robin scheme to access the N2HET RAM.
		3h	Reserved
20-19	Reserved	0	Reads return 0. Writes have no effect.
18	PPF		Protect Program Fields
			The PPF bit together with the Turn On/Off (TO) bit allows you to protect the program fields of all instructions in N2HET RAM.
			When TO = 0:
		0	All masters can read and write the program fields.
		1	All masters can read and write the program fields.
			When TO = 1:
		0	All masters can read and write the program fields.
		1	The program fields are readable but not writable for all masters, which could access the N2HET RAM. Possible masters are the CPU, HTU, and a secondary CPU (if available). Writes initiated by these masters are discarded.
17	IS		Ignore Suspend
		0	N2HET is stopped on suspend (the current timer instruction is completed). Timer RAM can be freely accessed during suspend.
		1	N2HET ignores suspend mode and continues operation.

Table 17-14. Global Configuration Register (HETGCR) Field Descriptions (continued)

Bit	Field	Value	Description
16	CMS		Clk_master/slave
			This bit is used to synchronize multi-N2HETs. If set (N2HET is master), the N2HET outputs a signal to synchronize the prescalers of the slave N2HET. By default, this bit is reset, which means a slave configuration.
			Note: This bit must be set to 1 for single-N2HET configuration.
		0	N2HET is configured as a slave.
		1	N2HET is configured as a master.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	TO		Turn On/Off
			TO does not affect the state of the pins. You must set/reset the timer pins when they are turned off, or re-initialize the timer RAM and control registers before a reset. After a device reset, the timer is turned off by default.
		0	N2HET is OFF. The timer program stops executing. Turn-off is automatically delayed until the current timer program loop is completed. Turn-off does not affect the content of the timer RAM, ALU registers, or control registers. Turn-off resets all flags.
		1	N2HET is ON. The timer program execution starts synchronously to the Loop clock. In case of multiple N2HETs configuration, the slave N2HETs are waiting for the loop clock to come from the master before starting execution. Then, the timer address points automatically address 00h (corresponding to program start).

17.4.2 Prescale Factor Register (HETPFR)

Figure 17-36. Prescale Factor Register (HETPFR) [offset = 04h]

31	16
Reserved	
R-0	

15	11	10	8	7	6	5	0
Reserved		LRPFC		Reserved		HRPFC	
R-0		R/WP-0		R-0		R/WP-0	

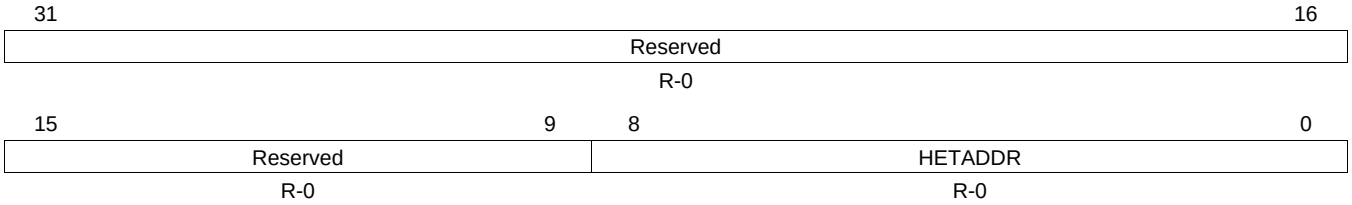
LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 17-15. Prescale Factor Register (HETPFR) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reads return 0. Writes have no effect.
10-8	LRPFC		Loop-Resolution Pre-scale Factor Code. LRPFC determines the loop-resolution prescale divide rate (lr).
		0	/1
		1h	/2
		2h	/4
		3h	/8
		4h	/16
		5h	/32
		6h	/64
		7h	/128
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	HRPFC		High-Resolution Pre-scale Factor Code. HRPFC determines the high-resolution prescale divide rate (hr).
		0	/1
		1h	/2
		2h	/3
		3h	/4
		:	:
		3Dh	/62
		3Eh	/63
		3Fh	/64

17.4.3 N2HET Current Address Register (HETADDR)

Figure 17-37. N2HET Current Address (HETADDR) [offset = 08h]



LEGEND: R = Read only; -*n* = value after reset

Table 17-16. N2HET Current Address (HETADDR) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	HETADDR		N2HET Current Address Read: Returns the current N2HET program address. Write: No effect.

17.4.4 Offset Index Priority Level 1 Register (HETOFF1)

Figure 17-38. Offset Index Priority Level 1 Register (HETOFF1) [offset = 0Ch]

31	16
Reserved	
R-0	

15	6	5	0
Reserved		OFFSET1	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-17. Offset Index Priority Level 1 Register (HETOFF1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	OFFSET1		OFFSET1 indexes the currently pending high-priority interrupt. Offset values and sources are listed in Table 17-18. Read: Read of these bits determines the pending N2HET interrupt. Write: No effect. **Note:** In any read operation mode, the corresponding flag (in the HETFLG) is also cleared. In Emulation mode the corresponding flag is not cleared.

Table 17-18. Interrupt Offset Encoding Format

Offset Value	Source No.
0	No interrupt
1	Instruction 0, 32, 64...
2	Instruction 1, 33, 65...
:	:
32	Instruction 31, 63, 95...
33	Program Overflow
34	APCNT Underflow
35	APCNT Overflow

17.4.5 Offset Index Priority Level 2 Register (HETOFF2)

Figure 17-39. Offset Index Priority Level 2 Register (HETOFF2) [offset = 10h]

31	16
Reserved	
R-0	

15	6	5	0
Reserved		OFFSET2	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 17-19. Offset Index Priority Level 2 Register (HETOFF2) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	OFFSET2		OFFSET2 indexes the currently pending low-priority interrupt. Offset values and sources are listed in Table 17-18. Read: Read of these bits determines the pending N2HET interrupt. Write: No effect. **Note:** In any read operation mode, the corresponding flag (in the HETFLG) is also cleared. In Emulation mode the corresponding flag is not cleared.

17.4.6 Interrupt Enable Set Register (HETINTENAS)

Figure 17-40. Interrupt Enable Set Register (HETINTENAS) [offset = 14h]

31	16
HETINTENAS	
R/W-0	

15	0
HETINTENAS	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-20. Interrupt Enable Set Register (HETINTENAS) Field Descriptions

Bit	Field	Value	Description
31-0	HETINTENAS[n]		Interrupt Enable Set bits. HETINTENAS is readable and writable in any operation mode. Writing a 1 to bit x enables the interrupts of the N2HET instructions at N2HET addresses x+0, x+32, x+64... Generating an interrupt requires to set bit x in HETINTENAS and to enable the interrupt bit in one of the instructions at addresses x+0, x+32, x+64, and so on. To avoid ambiguity, only one of the instructions x+0, x+32, x+64, and so on should have the interrupt enable bit (inside the instruction) set. Writing a 0 to HETINTENAS has no effect. When reading from HETINTENAS bit x gives the information, if N2HET instructions x+0, x+32, x+64... have the interrupt enabled or disabled.
		0	Read: Interrupt is disabled. Write: No effect.
		1	Read: Interrupt is enabled. Write: Interrupt will be enabled.

17.4.7 Interrupt Enable Clear Register (HETINTENAC)

Figure 17-41. Interrupt Enable Clear (HETINTENAC) [offset = 18h]

31	16
HETINTENAC	
R/W-0	

15	0
HETINTENAC	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-21. NHET Interrupt Enable Clear (HETINTENAC) Field Descriptions

Bit	Field	Value	Description
31-0	HETINTENAC[n]		Interrupt Enable Clear bits. HETINTENAC is readable and writable in any operation mode. Writing a 1 to bit x disables the interrupts of the N2HET instructions at N2HET addresses x+0, x+32, x+64, and so on. (See also description in Table 17-20). Writing a 0 to HETINTENAC has no effect. When reading from HETINTENAC bit x gives the information, if N2HET instructions x+0, x+32, x+64, and so on, have the interrupt enabled or disabled.
		0	Read: Interrupt is disabled. Write: No effect.
		1	Read: Interrupt is enabled. Write: Interrupt will be disabled.

17.4.8 Exception Control Register 1 (HETEXC1)

Figure 17-42. Exception Control Register (HETEXC1)

31	25	24
Reserved		APCNT_OVRFL_ENA
R-0		R/W-0

23	17	16
Reserved		APCNT_UNRFL_ENA
R-0		R/W-0

15	9	8
Reserved		PRGM_OVRFL_ENA
R-0		R/W-0

7	3	2	1	0
Reserved		APCNT_OVRFL_PRY	APCNT_UNRFL_PRY	PRGM_OVRFL_PRY
R-0		R/W-0	R/W-0	R/W-0

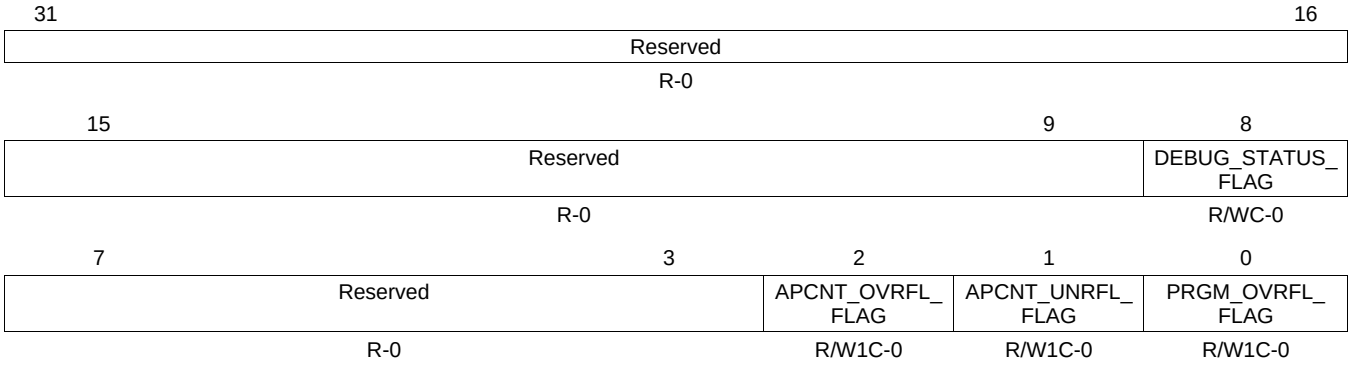
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-22. Exception Control Register 1 (HETEXC1) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
24	APCNT_OVRFL_ENA		APCNT Overflow Enable
		0	APCNT overflow exception is not enabled.
		1	Enables the APCNT overflow exception.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	APCNT_UNRFL_ENA		APCNT Underflow Enable
		0	APCNT underflow exception is not enabled.
		1	Enables the APCNT underflow exception.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	PRGM_OVRFL_ENA		Program Overflow Enable
		0	The program overflow exception is not enabled.
		1	Enables the program overflow exception.
7-3	Reserved	0	Reads return 0. Writes have no effect.
2	APCNT_OVRFL_PRY		APCNT Overflow Exception Interrupt Priority
		0	Exception priority level 2.
		1	Exception priority level 1.
1	APCNT_UNRFL_PRY		APCNT Underflow Exception Interrupt Priority
		0	Exception priority level 2.
		1	Exception priority level 1.
0	PRGM_OVRFL_PRY		ProgramOverflow Exception Interrupt Priority
		0	Exception priority level 2.
		1	Exception priority level 1.

17.4.9 Exception Control Register 2 (HETEXC2)

Figure 17-43. Exception Control Register 2 (HETEXC2)



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 17-23. Exception Control Register 2 (HETEXC2) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	DEBUG_STATUS_FLAG		Debug Status Flag. This flag is set when N2HET has stopped at a breakpoint. Also generates a debug request to halt the ARM CPU.
		0	Read: N2HET is either running, or stopped, flag cleared but not yet restarted. Write: No effect.
		1	Read: N2HET is stopped at a breakpoint. Write: Clears the bit. To restart N2HET clear this bit and then restart the ARM CPU. The N2HET and ARM CPU will start synchronously.
7-3	Reserved	0	Reads return 0. Writes have no effect.
2	APCNT_OVRFL_FLAG		APCNT Overflow Flag
		0	Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared. Write: Clears the bit.
1	APCNT_UNDFL_FLAG		APCNT Underflow Flag
		0	Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared. Write: Clears the bit.
0	PRGM_OVERFL_FLAG		Program Overflow Flag
		0	Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared Write: Clears the bit.

17.4.10 Interrupt Priority Register (HETPRY)

Figure 17-44. Interrupt Priority Register (HETPRY) [offset = 24h]

31	16
HETPRY	
R/WP-0	

15	0
HETPRY	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 17-24. Interrupt Priority Register (HETPRY) Field Descriptions

Bit	Field	Value	Description
31-0	HETPRY[n]		HET Interrupt Priority Level bits
			Used to select the priority of any of the 32 potential interrupt sources coming from N2HET instructions.
		0	Interrupt priority level 2 (low level).
		1	Interrupt priority level 1 (high level).

17.4.11 Interrupt Flag Register (HETFLG)

Figure 17-45. Interrupt Flag Register (HETFLG) [offset = 28h]

31	16
HETFLAG	
R/W1C-0	

15	0
HETFLAG	
R/W1C-0	

LEGEND: R/W = Read/Write; W1C = Write 1 to clear; X = Unknown; -*n* = value after reset

Table 17-25. Interrupt Flag Register (HETFLG) Field Descriptions

Bit	Field	Value	Description
31-0	HETFLAG[n]		Interrupt Flag Register Bits
			Bit x is set when an interrupt condition has occurred on one of the instructions x+0, x+32, x+64, and so on. The flag position x (in the register) is decoded from the five LSBs of the instruction address that generated the interrupt. The hardware will set the flag only if the interrupt enable bit (in the corresponding instruction) is set. The flag will be set even if bit x in the Interrupt Enable Set Register (HETINTENAS) is not enabled. Enabling bit x in HETINTENAS is required if an interrupt should be generated.
			Clearing the flag can be done by writing a 1 to the flag. Alternatively reading the corresponding Offset Index Priority Level 1 Register (HETOFF1) or Offset Index Priority Level 2 Register (HETOFF2) will automatically clear the flag.
		0	Read: No N2HET instruction with an interrupt has been reached since the flag was cleared.
			Write: No effect.
		1	Read: A N2HET instruction with an interrupt has been reached since the flag was cleared.
			Write: Clears the bit.

17.4.12 AND Share Control Register (HETAND)

Figure 17-46. AND Share Control Register (HETAND) [offset = 2Ch]

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
AND SHARE31/30	AND SHARE29/28	AND SHARE27/26	AND SHARE25/24	AND SHARE23/22	AND SHARE21/20	AND SHARE19/18	AND SHARE17/16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AND SHARE15/14	AND SHARE13/12	AND SHARE11/10	AND SHARE9/8	AND SHARE7/6	AND SHARE5/4	AND SHARE3/2	AND SHARE1/0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-26. AND Share Control Register (HETAND) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	ANDSHARE n+1 / n		AND Share Enable Enable the AND sharing of the same pin for two HR structures. For example, if bit ANDSHARE1/0 is set, the pin HET[0] will then be commanded by a logical AND of both HR structures 0 and 1. **Note:** If HR AND SHARE bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		0	HR Output of HET[n+1] and HET[n] are not AND shared.
		1	HR Output of HET[n+1] and HET[n] are AND shared onto pin HET[n].

17.4.13 HR Share Control Register (HETHRSH)

Figure 17-47. HR Share Control Register (HETHRSH) [offset = 34h]

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
HR SHARE31/30	HR SHARE29/28	HR SHARE27/26	HR SHARE25/24	HR SHARE23/22	HR SHARE21/20	HR SHARE19/18	HR SHARE17/16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
HR SHARE15/14	HR SHARE13/12	HR SHARE11/10	HR SHARE9/8	HR SHARE7/6	HR SHARE5/4	HR SHARE3/2	HR SHARE1/0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-27. HR Share Control Register (HETHRSH) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HRSHARE n+1 / n		HR Share Bits Enables the share of the same pin for two HR structures. For example, if bit HRSHARE1/0 is set, the pin HET[0] will then be connected to both HR input structures 0 and 1. **Note:** If HR share bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		0	HR Input of HET[n+1] and HET[n] are not shared.
		1	HR Input of HET[n+1] and HET[n] are shared; both measure pin HET[n].

17.4.14 XOR Share Control Register (HETXOR)

Figure 17-48. XOR Share Control Register (HETXOR) [offset = 38h]

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
XOR SHARE31/30	XOR SHARE29/28	XOR SHARE27/26	XOR SHARE25/24	XOR SHARE23/22	XOR SHARE21/20	XOR SHARE19/18	XOR SHARE17/16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
XOR SHARE15/14	XOR SHARE13/12	XOR SHARE11/10	XOR SHARE9/8	XOR SHARE7/6	XOR SHARE5/4	XOR SHARE3/2	XOR SHARE1/0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-28. XOR Share Control Register (HETXOR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	XORSHARE n+1 / n		XOR Share Enable Enable the XOR-share of the same pin for two output HR structures. For example, if bit XORSHARE1/0 is set, the pin HET[0] will then be commanded by a logical XOR of both HR structures 0 and 1. **Note:** If XOR share bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		0	HR Output of HET[n+1] and HET[n] are not XOR shared.
		1	HR Output of HET[n+1] and HET[n] are XOR shared onto pin HET[n].

17.4.15 Request Enable Set Register (HETREQENS)

Figure 17-49. Request Enable Set Register (HETREQENS) [offset = 3Ch]

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
REQENA7	REQENA6	REQENA5	REQENA4	REQENA3	REQENA2	REQENA1	REQENA0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-29. Request Enable Set Register (HETREQENS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	REQENAn		Request Enable Bits
		0	Read: Returns the information that request line n is disabled. Write: No effect.
		1	Read: Returns the information that request line n is enabled. Write: Writing a 1 to bit n enables the N2HET request line n. **Note:** The request line triggers an HTU double control packet (DCP). **Note:** A disabled request line does not memorize old requests. So there are no pending requests to service after enabling request line n.

17.4.16 Request Enable Clear Register (HETREQENC)

Figure 17-50. Request Enable Clear Register (HETREQENC) [offset = 40h]

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
REQDIS7	REQDIS6	REQDIS5	REQDIS4	REQDIS3	REQDIS2	REQDIS1	REQDIS0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-30. Request Enable Clear Register (HETREQENC) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	REQDISn		Request Disable Bits
		0	Read: Returns the information that request line n is disabled. Write: No effect.
		1	Read: Returns the information that request line n is enabled. Write: Writing a 1 to bit n disables the N2HET request line n.

NOTE: Please refer to the device data manual for information about how each of the 8 N2HET request lines are connected to these modules. See also Section 17.2.9.

17.4.17 NHET Direction Register (HETDIR)

Figure 17-51. N2HET Direction Register (HETDIR) [offset = 4Ch]

31	16
HETDIR	
R/W-0	

15	0
HETDIR	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-31. N2HET Direction Register (HETDIR) Field Descriptions

Bit	Field	Value	Description
31-0	HETDIR[n]		Data direction of NHET pins
		0	Pin HET[n] is an input (and its output buffer is tristated).
		1	Pin HET[n] is an output.

NOTE: Table 17-9 shows how the register bits of DIR, PULDIS, and PULSEL are affecting the N2HET pins.

17.4.18 N2HET Data Input Register (HETDIN)

Figure 17-52. N2HET Data Input Register (HETDIN) [offset = 50h]

31	16
HETDIN	
R-x	

15	0
HETDIN	
R-x	

LEGEND: R = Read only; -x = value is unknown after reset; -*n* = value after reset

Table 17-32. N2HET Data Input Register (HETDIN) Field Descriptions

Bit	Field	Value	Description
31-0	HETDIN[n]		Data input. This bit displays the logic state of the pin.
		0	Pin HET[n] is at logic low (0).
		1	Pin HET[n] is at logic high (1).

17.4.19 N2HET Data Output Register (HETDOUT)

Figure 17-53. N2HET Data Output Register (HETDOUT) [offset = 54h]

31	16
HETDOUT	
R/W-0	

15	0
HETDOUT	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-33. N2HET Data Output Register (HETDOUT) Field Descriptions

Bit	Field	Value	Description
31-0	HETDOUT[n]		Data out write. Writes to this bit will only take effect when the pin is configured as an output. The current logic state of the pin will be displayed by this bit even when the pin state is changed by writing to HETDSET or HETDCLR.
		0	Pin HET[n] is at logic low (0).
		1	Pin HET[n] is at logic high (1) if the HETPDR[n] bit = 0 or the output is in high-impedance state if the HETPDR[n] bit = 1.

17.4.20 NHET Data Set Register (HETDSET)

Figure 17-54. N2HET Data Set Register (HETDSET) [offset = 58h]

31	16
HETDSET	
R/WS-0	

15	0
HETDSET	
R/WS-0	

LEGEND: R/W = Read/Write; S = Set; -*n* = value after reset

Table 17-34. N2HET Data Set Register (HETDSET) Field Descriptions

Bit	Field	Value	Description
31-0	HETDSET[n]		This register allows bits of HETDOUT to be set while avoiding the pitfalls of a read-modify-write sequence in a multitasking environment. Bits written as a logic 1 set the same bit in the HETDOUT register; while bits written as logic 0 leave the same bit in HETDOUT unchanged. Reads from this address return the value of the HETDOUT register.
		0	Write: HETDOUT[n] is unchanged.
		1	Write: HETDOUT[n] is set.

17.4.21 N2HET Data Clear Register (HETDCLR)

Figure 17-55. N2HET Data Clear Register (HETDCLR) [offset = 5Ch]

31	16
HETDCLR	
R/WC-0	

15	0
HETDCLR	
R/WC-0	

LEGEND: R/W = Read/Write; C = Clear; -*n* = value after reset

Table 17-35. N2HET Data Clear Register (HETDCLR) Field Descriptions

Bit	Field	Value	Description
31-0	HETDCLR[n]		This register allows bits of HETDOUT to be cleared while avoiding the pitfalls of a read-modify-write sequence in a multitasking environment. Bits written as a logic 1 clear the same bit in the HETDOUT register; while bits written as logic 0 leave the same bit in HETDOUT unchanged. Reads from this address return the value of the HETDOUT register.
		0	Write: HETDOUT[n] is unchanged.
		1	Write: HETDOUT[n] is cleared.

17.4.22 N2HET Open Drain Register (HETPDR)

Values in this register enable or disable the open drain capability of the data pins.

Figure 17-56. N2HET Open Drain Register (HETPDR) [offset = 60h]

31	16
HETPDR	
R/W-0	

15	0
HETPDR	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-36. N2HET Open Drain Register (HETPDR) Field Descriptions

Bit	Field	Value	Description
31-0	HETPDR[n]		Open drain control for HET[n] pins
		0	The pin is configured in push/pull mode.
		1	The pin is configured in open drain mode. The HETDOUT register controls the state of the output buffer:
			HETDOUT[n] = 0 The output buffer of pin HET[n] is driven low.
			HETDOUT[n] = 1 The output buffer of pin HET[n] is tristated.

17.4.23 N2HET Pull Disable Register (HETPULDIS)

Values in this register enable or disable the pull-up/-down functionality of the pins.

Figure 17-57. N2HET Pull Disable Register (HETPULDIS) [offset = 64h]

31	16
HETPULDIS	
R/W-n	

15	0
HETPULDIS	
R/W-n	

LEGEND: R/W = Read/Write; -*n* = value after reset and is device dependent, see device-specific data manual

Table 17-37. N2HET Pull Disable Register (HETPULDIS) Field Descriptions

Bit	Field	Value	Description
31-0	HETPULDIS[n]		Pull disable for N2HET pins.
		0	The pull functionality is enabled on pin HET[n].
		1	The pull functionality is disabled on pin HET[n].

NOTE: See device data manual for which pins provide programmable pullups/pulldowns.

Table 17-9 shows how the register bits of HETDIR, HETPULDIS and HETPSL are affecting the N2HET pins.

17.4.24 N2HET Pull Select Register (HETPSL)

Values in this register select the pull-up or pull-down functionality of the pins.

Figure 17-58. N2HET Pull Select Register (HETPSL) [offset = 68h]

31	16
HETPSL	
R/W-0	

15	0
HETPSL	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-38. N2HET Pull Select Register (HETPSL) Field Descriptions

Bit	Field	Value	Description
31-0	HETPSL[n]		Pull select for NHET pins.
		0	The pull down functionality is enabled if corresponding bit in HETPULDIS is 0.
		1	The pull up functionality is enabled if corresponding bit in HETPULDIS is 0.

NOTE: See device data manual for which pins provide programmable pullups/pulldowns.

Table 17-9 shows how the register bits of HETDIR, HETPULDIS, and HETPSL are affecting the N2HET pins.

The information of this register is also used to define the pin states after a parity error:

After a parity error all N2HET pins, which are

1. Defined as output pins in the HETDIR register
2. Not defined as open drain pins (with the HETPDR register)
3. Selected with the HETPPR register, will remain outputs, but automatically change their levels in the following way:
   - If the HETPSL register specifies 0 for the pin, it will switch to low level.
   - If the HETPSL register specifies 1 for the pin, it will switch to high level.

This behavior is independent of the value, which register HETPULDIS specifies for the corresponding pin.

17.4.25 Parity Control Register (HETPCR)

Figure 17-59. Parity Control Register (HETPCR) [offset = 74h]

31	16
Reserved	
R-0	

15	9	8	7	4	3	0
Reserved		TEST	Reserved		PARITY_ENA	
R-0		R/WP-0	R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

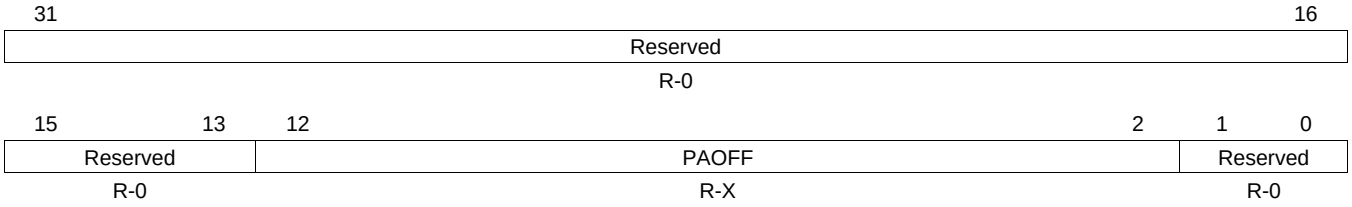
Table 17-39. Parity Control Register (HETPCR) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	TEST		Test Bit. When this bit is set, the parity bits are mapped into the peripheral RAM frame to make them accessible by the CPU.
		0	Read: Parity bits are not memory-mapped. Write: Disable mapping.
		1	Read: Parity bits are memory-mapped. Write: Enable mapping.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PARITY_ENA		Enable/disable parity checking. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected the N2HET_UERR signal is activated.
		5h	Read: Parity check is disabled. Write: Disable checking.
		Others	Read: Parity check is enabled. Write: Enable checking.

NOTE: It is recommended to write Ah to enable error detection, to guard against soft errors flipping PARITY_ENA to a disable state.

17.4.26 Parity Address Register (HETPAR)

Figure 17-60. Parity Address Register (HETPAR) [offset = 78h]



LEGEND: R = Read only; X = Value is unchanged after reset; -*n* = value after reset

Table 17-40. Parity Address Register (HETPAR) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12-2	PAOFF		Parity Error Address Offset. This register holds the offset address of the first parity error, which is detected in N2HET RAM. This error address is frozen from being updated until it is read by the CPU. During emulation mode, this address is frozen even when read. In case of a N2HET RAM parity error, PAOFF will contain the offset address of the erroneous 32-bit N2HET RAM field counted from the beginning of the N2HET RAM. Examples: The 32-bit program field of instruction 0 will return 0, the 32-bit control field of instruction 0 will return 1, ..., the 32-bit control field of instruction 1 will return 5, and so on. Read: Returns the offset address of the erroneous 32-bit word in bytes from the beginning of the N2HET RAM. Write: No effect.
1-0	Reserved	0	Reads return 0. Writes have no effect.

NOTE: The Parity Error Address Register will not be reset, neither by PORRST nor by any other reset source.

17.4.27 Parity Pin Register (HETPPR)

Figure 17-61. Parity Pin Register (HETPPR) [offset = 7Ch]

31	16
HETPPR	
R/W-0	

15	0
HETPPR	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-41. Parity Pin Register (HETPPR) Field Descriptions

Bit	Field	Value	Description
31-0	HETPPR[n]		NHET Parity Pin Select Bits. Allows HET[n] pins to be configured to drive to a known state when an N2HET parity error is detected.
		0	Pin HET[n] is not affected by the detection of an N2HET parity error.
		1	Pin HET[n] is driven to a known state when an N2HET parity error is detected. The known state is a function of bits HETDIR[n], HETPSL[n], HETPDR[n] as described in Table 17-42 (this state is also independent of HETPULDIS[n]).

Table 17-42. Known State on Parity Error

HETDIR[n]	HETPDR[n]	HETPSL[n]	Known State on Parity Error
0	x	x	High Impedance
1	0	0	Drive Logic 0
1	0	1	Drive Logic 1
1	1	x	High Impedance

17.4.28 Suppression Filter Preload Register (HETSFPRLD)

Figure 17-62. Suppression Filter Preload Register (HETSFPRLD) [offset = 80h]

31	18	17	16
Reserved		CCDIV	
R-0		R/W-0	

15	10	9	0
Reserved		CPRLD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17-43. Suppression Filter Preload Register (HETSFPRLD) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reads return 0. Writes have no effect.
17-16	CCDIV		Counter Clock Divider
			CCDIV determines the ratio between the counter clock and VCLK2.
		0	CCLK = VCLK2
		1h	CCLK = VCLK2 / 2
		2h	CCLK = VCLK2 / 3
		3h	CCLK = VCLK2 / 4
15-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	CPRLD		Counter Preload Value
			CPRLD contains the preload value for the counter clock.

17.4.29 Suppression Filter Enable Register (HETSFENA)

Figure 17-63. Suppression Filter Enable Register (HETSFENA) [offset = 84h]

31	16
HETSFENA	
R/W-0	

15	0
HETSFENA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-44. Suppression Filter Enable Register (HETSFENA) Field Descriptions

Bit	Field	Value	Description
31-0	HETSFENA[n]		Suppression Filter Enable Bits
			Note: If the pin is configured as an output by the N2HET Direction Register (HETDIR), the filter is automatically disabled independent on the bit in HETSFENA.
		0	The input noise suppression filter for pin HET[n] is disabled.
		1	The input noise suppression filter for pin HET[n] is enabled.

17.4.30 Loop Back Pair Select Register (HETLBPSEL)

Refer to Section 17.2.5.7 for a description of loopback test functions.

Figure 17-64. Loop Back Pair Select Register (HETLBPSEL) [offset = 8Ch]

31	30	29	28	27	26	25	24
LBPTYPE31/30	LBPTYPE29/28	LBPTYPE27/26	LBPTYPE25/24	LBPTYPE23/22	LBPTYPE21/20	LBPTYPE19/18	LBPTYPE17/16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
LBPTYPE15/14	LBPTYPE13/12	LBPTYPE11/10	LBPTYPE9/8	LBPTYPE7/6	LBPTYPE5/4	LBPTYPE3/2	LBPTYPE1/0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
LBPSEL31/30	LBPSEL29/28	LBPSEL27/26	LBPSEL25/24	LBPSEL23/22	LBPSEL21/20	LBPSEL19/18	LBPSEL17/16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
LBPSEL15/14	LBPSEL13/12	LBPSEL11/10	LBPSEL9/8	LBPSEL7/6	LBPSEL5/4	LBPSEL3/2	LBPSEL1/0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-45. Loop Back Pair Select Register (HETLBPSEL) Field Descriptions

Bit	Field	Value	Description
31-16	LBPTYPE n+1 / n		Loop Back Pair Type Select Bits These bits are valid only when IODFT mode is enabled (HETLBPDIR[19:16] = 1010).
		0	Digital loopback is selected for HR structures on pins HET[n+1] and HET[n].
		1	Analog loopback is selected for HR structures on pins HET[n+1] and HET[n].
15-0	LBPSEL n+1 / n		Loop Back Pair Select Bits These bits are valid only when IODFT mode is enabled (HETLBPDIR[19:16] = Ah). If bit x is set, the HR structures on pins HET[n+1] and HET[n] are connected in a loop back mode. The direction is given by LBPDIR n+1/n and type is selected by LBPTYPE n+1/n. The pin which is not driven by the N2HET pin actions can still be used as normal GIO pin.

17.4.31 Loop Back Pair Direction Register (HETLBPDIR)

Refer to Section 17.2.5.7 for a description of loopback test functions.

Figure 17-65. Loop Back Pair Direction Register (HETLBPDIR) [offset = 90h]

31							20	19			16
Reserved								LBPTSTENA			
R-0								R/WP-5h			
15	14	13	12	11	10	9	8				
LBPDIR31/30	LBPDIR29/28	LBPDIR27/26	LBPDIR25/24	LBPDIR23/22	LBPDIR21/20	LBPDIR19/18	LBPDIR17/16				
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0				
7	6	5	4	3	2	1	0				
LBPDIR15/14	LBPDIR13/12	LBPDIR11/10	LBPDIR9/8	LBPDIR7/6	LBPDIR5/4	LBPDIR3/2	LBPDIR1/0				
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0				

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 17-46. Loop Back Pair Direction Register (HETLBPDIR) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	LBPTSTENA		Loopback Test Enable Key
		5h	Loopback Test is disabled.
		Ah	Loopback Test is enabled.
		Others	Loopback Test is disabled.
15-0	LBPDIR n+1 / n		Loop Back Pair Direction Bits
		0	The HR structures on pins HET[n+1] and HET[n] are internally connected with HET[n] as input and HET[n+1] as output.
		1	The HR structures on pins HET[n+1] and HET[n] connected with HET[n] as output and HET[n+1] as input.

NOTE: The loop back direction can be selected independent on the HETDIR register setting.

17.4.32 N2HET Pin Disable Register (HETPINDIS)

Figure 17-66. N2HET Pin Disable Register (HETPINDIS) [offset = 94h]

31	16
HETPINDIS	
R/W-0	

15	0
HETPINDIS	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 17-47. NHET Pin Disable Register (HETPINDIS) Field Descriptions

Bit	Field	Value	Description
31-0	HETPINDIS[n]		N2HET Pin Disable Bits
		0	Logic low: No affect on the output buffer enable of the pin (is controlled by the value of the HETDIR[n] bit).
		1	Logic high: Output buffer of the pin is enabled if pin nDIS = 1, HET_PIN_ENA = 1, and HETDIR = 1; or disabled if nDIS = 0, HETDIR = 0, or HET_PIN_ENA = 0.

17.5 Instruction Set

17.5.1 Instruction Summary

Table 17-48 presents a list of the instructions in the N2HET instruction set. The pages following describe each instruction in detail.

Table 17-48. Instruction Summary

Abbreviation	Instruction Name	Opcode	Sub-Opcode	Cycles[1]
ACMP	Angle Compare	Ch	-	1
ACNT	Angle Count	9h	-	2
ADCNST	Add Constant	5h	-	2
ADC	Add with Carry and Shift	4h	C[25:23] = 011, C5 = 1	1-3
ADD	Add and Shift	4h	C[25:23] = 001, C5 = 1	1-3
ADM32	Add Move 32	4h	C[25:23] = 000, C5 = 1	1-2
AND	Bitwise AND and Shift	4h	C[25:23] = 010, C5 = 1	1-3
APCNT	Angle Period Count	Eh	-	1-2
BR	Branch	Dh	-	1
CNT	Count	6h	-	1-2
DADM64	Data Add Move 64	2h	-	2
DJZ	Decrement and Jump if -zero	Ah	P[7:6] = 10	1
ECMP	Equality Compare	0h	C[6:5] = 00	1
ECNT	Event Count	Ah	P[7:6] = 01	1
MCMP	Magnitude Compare	0h	C[6] = 1	1
MOV32	Move 32	4h	C[5] = 0	1-2
MOV64	Move 64	1h	-	1
OR	Bitwise OR	4h	C[25:23] = 100, C5 = 1	1-3
PCNT	Period/Pulse Count	7h	-	1
PWCNT	Pulse Width Count	Ah	P[7:6] = 11	1
RADM64	Register Add Move 64	3h	-	1
RCNT	Ratio Count	Ah	P[7:6] = 00, P[0] = 1	3
SBB	Subtract with Borrow and Shift	4h	C[25:23] =110, C[5] = 1	1-3
SCMP	Sequence Compare	0h	C[6:5] = 01	1
SCNT	Step Count	Ah	P[7:6] = 00, P[0] = 0	3
SHFT	Shift	Fh	C[3] = 0	1
SUB	Subtract and Shift	4h	C[25:23] = 101, C[5] = 1	1-3
WCAP	Software Capture Word	Bh	-	1
WCAPE	Software Capture Word and Event Count	8h	-	1
XOR	Bitwise Exclusive-Or and Shift	4h	C[25:23] = 111, C[5] = 1	1-3

(1) Cycles refers to the clock cycle of the N2HET module; which on most devices is VCLK2. (Check the device datasheet description of clock domains to confirm). If the high-resolution prescale value is set to /1, then this is also the same as the number of HR clock cycles.

Table 17-49. FLAGS Generated by Instruction

Abbreviation	Flag Name	Set/Reset by	Used by
C	Carry Flag	ADC, ADD, AND, OR, RCNT, SBB, SUB, XOR	ADC, BR, SBB
N	Negative Flag	ADC, ADD, AND, OR, SBB, SUB, XOR	BR
V	Overflow Flag	ADC, ADD, AND, OR, SBB, SUB, XOR	BR
Z	Zero flag	ACNT, ADC, ADD, AND, APCNT, CNT, OR, PCNT, SBB, SCNT, SHFT, SUB, XOR	ACMP, ACNT, BR, ECMP, MCMP, MOV32, RCNT, SCMP, SHFT
X	Angle Compare Match Flag	ACMP	SCMP
SWF 0-1	Step Width flags	SCNT	ACNT
NAF	New Angle Flag	ACNT	NAF_global
NAF_global	New Angle Flag (global)	HWAG or NAF	ACMP, BR, CNT, ECMP, ECNT
ACF	Acceleration Flag	ACNT	,ACNT, SCNT
DCF	Deceleration Flag	ACNT	,ACNT, SCNT
GPF	Gap Flag	ACNT	ACNT, APCNT

The instructions capable of generating software interrupts are listed in Table 17-50.

Table 17-50. Interrupt Capable Instructions

Interrupt Capable Instructions	Non Interrupt Capable Instructions
ACMP	ADC
ACNT	ADCNST
APCNT	ADD
BR	ADM32
CNT	AND
DJZ	DADM32
ECMP	MOV32
ECNT	MOV64
MCMP	OR
PCNT	RADM64
PWCNT	RCNT
SCMP	SBB
SHFT	SCNT
WCAP	SUB
WCAPE	XOR

17.5.2 Abbreviations, Encoding Formats and Bits

Abbreviations marked with a star (*) are available only on specific instructions.

U Reading a bit marked with U will return an indeterminate value.

BRK Defines the software breakpoint for the device software debugger.
Default: OFF
Location: Program field [22]

next Defines the program address of the next instruction in the program flow. This value may be a label or an 9-bit unsigned integer.
Default: Current instruction + 1
Location: Program field [21:13]

reqnum* Defines the number of the request line (0,1,..,7) to trigger the HTU.
Default: 0
Location: Program field [25:23]

request* Allows to select between no request (NOREQ), request (GENREQ) and quiet request (QUIET). See Section 17.2.9.
Default: No request
Location: Control Field [28:27]

Request	C[28	]C[27]	To HTU
NOREQ	0	0	no request
	1	0	
GENREQ	0	1	request
QUIET	1	1	quiet request

remote* Determines the 9-bit address of the remote address for the instruction.
Default: Current instruction + 1
Location: Program field [8:0]

control Determines whether the immediate data field [31:0] is cleared when it is read. When the bit is not set, reads do not clear the immediate data field.
Default: OFF
Location: Control field [26]

en_pin_action* Determines whether the selected pin is ON so that the action occurs on the chosen pin
Default: OFF
Location: Control field [22]

Cond_addr* Conditional address (optional): Defines the address of the next instruction when the condition occurs.
Default: Current address + 1
Location: Control field [21:13]

Pin* Pin Select: Selects the pin on which the action occurs. Enter the pin number.
Default: pin 0
Location: Control field [12:8] except PCNT

The format CC{pin number} is also supported.

MSB				LSB	Description
0	0	0	0	0	Select HET[0]
0	0	0	0	1	Select HET[1]
(Each pin may be selected by writing its number in binary)					
1	1	1	1	0	Select HET[30]
1	1	1	1	1	Select HET[31]

Reg* Register select: Selects the register for data comparison and storage

Default: No register (None)

Location: Control field [2:1] except for CNT instruction.

Extended Register Select C[7] is available for ACMP, ADC, ADD, ADM32, AND, DADM64, ECMP, ECNT, MCMP, MOV32, MOV64, OR, RADM64, SBB, SHFT, SUB, WCAP, WCAPE instructions.

Register	Ext Reg. C[7]	C[2]	C[1]
A	0	0	0
B	0	0	1
T	0	1	0
None	0	1	1
R	1	0	0
S	1	0	1
Reserved	1	1	0
Reserved	1	1	1

Action* (2 Action Option) Either sets or clears the pin

Default: Clear

Location: Control Field [4]

Action	C[4]
Clear	0
Set	1

Action* (4 Action Option) Either sets, clears, pulse high or pulse low on the pin. Set/clear are single pin actions, pulse high/low include the opposite pin action.

Default: Clear

Location: Control Field [4:3]

Action	Action Type	C[4]	C[3]
Clear	Set low on match	0	0
Set	Set high on match	1	0
Pulse Low	Set low on match + reset to high on Z=1 (opposite action)	0	1
Pulse High	Set high on match + reset to low on Z=1 (opposite action)	1	1

hr_lr* Specifies HIGH/LOW data resolution. If the hr_lr field is HIGH, the instruction uses the hr_data field. If the hr_lr field is LOW, the hr_data field is ignored.

Default: HIGH

Location: Program Field [8]

hr_lr	Prog. field [8]
LOW	1
HIGH	0

prv* Specifies the initial value defining the previous bit (see Section 17.2.5.8). A value of ON sets the previous pin-level bit to 1. A value of OFF sets the initial value of the previous (prv) bit to 0. The prv bit is overwritten (set or reset) by the N2HET the first time the instruction is executed.

Default: OFF

Location: Control Field [25]

cntl_val* Available for DADM64, MOV64, and RADM64, this bit field allows the user to specify the replacement value for the remote control field.

comp_mode* Specifies the compare mode. This field is used with the 64-bit move instructions. This field ensures that the sub-opcodes are moved correctly.

Default: ECMP

Location: Control Field [6:5]

Action	C[6]	C[5]	Order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

17.5.3 Instruction Description

The following sections provide information for individual instructions.

Parameters in [] are optional. Refer to the N2HET assembler user guide for the default values when parameters are omitted.

17.5.3.1 ACMP (Angle Compare)

Syntax

```
ACMP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin={pin number}
[action={CLEAR | SET}]
reg={A | B | R | S | T | NONE}
[irq ={OFF | ON}]
data={25-bit unsigned integer}
}
```

Figure 17-67. ACMP Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Request Number	BRK	Next program address	1100	Reserved
6	3	1	9	4	9

Figure 17-68. ACMP Control Field (C31:C0)

31–29	28–27	26	25	24–23	22	21–16
Reserved	Request type	Control	Cout prv	Reserved	En. pin action	Conditional address
3	2	1	1	2	1	9

15–13	12–8	7	6–5	4	3	2–1	0
Conditional address	Pin select	Ext. Reg	Reserved	Pin action	Res.	Register select	Int. ena
9	5	1	2	1	1	2	1

Figure 17-69. ACMP Data Field (D31:D0)

31–7	6–0
Data	Reserved
25	7

Cycles One

Register modified Selected register (A, B, R, S, or T)

The purpose of the comparison is to assert pin action when the angle compare value lies between the old counter value and the new counter value (held in the selected register). Since the angle increment varies from one loop resolution clock to another, an exact equality test cannot be applied. Instead, the following inequality is used to determine the occurrence of a match:

Old counter value < *Angle compare value* ≤ *New counter value*

This is done by performing following comparisons:

Selected register value minus angle increment < angle compare value
Angle compare value ≤ Selected register value

register	Register B is recommended for typical applications with ACMP.
irq	Specifies whether or not an interrupt is generated. Specifying ON generates an interrupt when the edge state is satisfied and the gap flag is set. Specifying OFF prevents an interrupt from being generated. Default: OFF.
data	Specifies the 25-bit angle compare value.

Execution

```
X = 0;
If (Data <= Selected Register)
      Cout = 0;
else
      Cout = 1;
If (Z == 0 AND (Selected Register - Angle Inc. < Data ) AND Cout == 0) OR

 (Z == 1 AND (Cout_prv == 1 OR Cout == 0)))
{
      X = 1;
      If (Enable Pin Action == 1)
          Selected Pin = Pin Action AT next loop resolution clock;

      If (Interrupt Enable == 1)
          HETFLG[n] = 1;   /* n depends on address */
      If ([C28:C27] == 01)
          Generate request on request line [P25:P23];
      If ([C28:C27] == 11)
          Generate quiet request on request line [P25:P23];

      Jump to Conditional Address;
 }

 else
      Jump to Next Program Address;

Cout_prv = Cout (always executed)
```

NOTE: Carry-Out Signal (Cout)

Cout is the carry-out signal of the adder. Even if it is not a flag, it is valid all along ACMP instruction execution.

Angle inc. = NAF_global or hardware angle generator 11-bit input.

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.2 ACNT (Angle Count)

Syntax

```
ACNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
edge={RISING | FALLING}
[irq ={OFF | ON}]
[control={OFF | ON}]
[prv={OFF | ON}]
gapend ={25-bit unsigned integer}
data={25-bit unsigned integer}
}
```

Figure 17-70. ACNT Program Field (P31:P0)

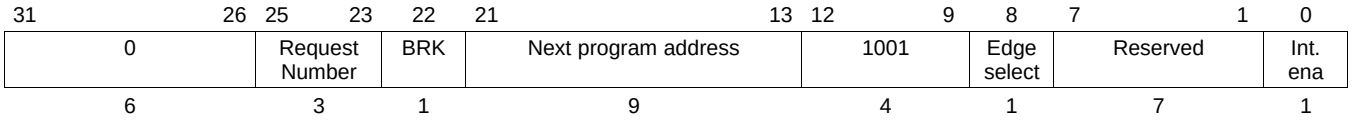


Figure 17-71. ACNT Control Field (C31:C0)

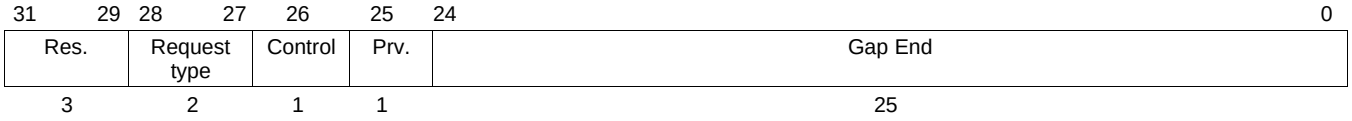
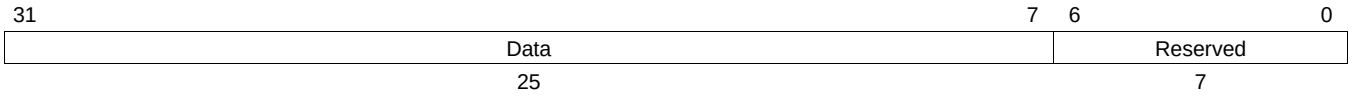


Figure 17-72. ACNT Data Field (D31:D0)



Cycles

Two, as follows:

- First cycle: Angle increment condition and gap end comparison.
- Second cycle: Gap start comparison.

Register modified

Register B (angle value)

Description

This instruction defines a specialized virtual timer used after SCNT and APCNT to generate an angle-referenced time base that is synchronized to an external signal (that is, a toothed wheel signal). ACNT uses pin HET[2] exclusively. The edge select must be the same as the HET[2] edge which was selected in the previous APCNT.

ACNT refers to the same step width selection that the previous SCNT saved in flags SWF0 and SWF1 (see information on SCNT).

ACNT detects period variations of the external signal measured by APCNT and compensates related count errors.

A period increase is flagged in the deceleration flag (DCF). A period decrease is flagged in the acceleration flag (ACF). If no variation is detected, ACNT increments the counter value each time SCNT reaches its target.

If acceleration is detected, ACNT increments the counter value on each timer resolution. If deceleration is detected ACNT does not increment and is thus saturated.

ACNT also specifies the gap end angle value defining the end value of a gap range in ACNT where period measurements in APCNT are temporarily stopped to mask singularities in the external signal. ACNT uses register A containing gap start and register B to store the counter value.

Edge Specifies the edge for the input capture pin (HET[2]).

Action	P8	Edge Select
Rising	1	Detects a rising edge of HET[2]
Falling	0	Detects a falling edge of HET[2]

irq ON generates an interrupt when the edge state is satisfied and the gap flag is set. OFF prevents an interrupt from being generated.

Default: OFF.

gapend Defines the 25-bit end value of a gap range. The start value is defined in the SCNT instruction.

GAPEND = (Step Value * (# of teeth on the toothed wheel + # of missing teeth)) - 1

data Specifies the 25-bit initial count value for the data field.

Default: 0.

NOTE: Target Edge Field

The target edge field represents the three LSBs of data field register in case of step width = 8, four LSBs for step width = 16, five LSBs for step width = 32 and six LSBs for step width = 64.

Execution

```
Increment Condition: ((Z = 1 AND DCF = 0) OR ACF = 1)
Pin Edge Condition: Specified edge detected on HET[2]
Target Edge Condition: (Target Edge field in data field = 0) AND (Angle
Increment condition is true) AND (GPF = 0)

If (Angle Increment Condition) is false
{
      NAF = 0;
      Register B = Data field register;
}
else
{
      NAF = 1;
      If (Counter value != GapEnd)
      {
           Register B = Data field register + 1;
           Data Field Register = Counter value + 1;
      }
```

```
      else
      {
           Register B = 0;
           Data Field Register = 0;
           If (ACF == 0) DCF = 1;
      }
}

Z = 0;

If (Data field register == GapStart)
{
       GPF = 1;
       If (Target Edge condition is true)
       {
           ACF = 0;
           If ((specified edge is not detected on pin HET[2]) AND (data
           field register != 0) AND (ACF == 0) AND (angle increment condition
           is true))
             DCF = 1;
       }
       If (specified edge is detected on pin HET[2])
       {
           DCF = 0;
           If ((target_edge_field != 0) AND (DCF == 0)) ACF = 1;
           If (GPF == 1)
           {
             GPF = 0;
             Z = 1;
             If (Interrupt Enable == 1)
                HETFLG[n] = 1;      /* n depends on address */
             If ([C28:C27] == 01)
                Generate request on request line [P25:P23];;
             If ([C28:C27] == 11)
                Generate quiet request on request line
                [P25:P23];
           }
        }
}

If ((target_edge_field != 0) and (pin_edge_cond == 1))
{
   pin_update = 0;
}
else if (target_edge_field == 0)
{
   pin_update = 1;
}

If (pin_update is true in next loop clock cycle)
{
   Prv bit = Current Lx value of HET[2] pin;
}

Jump to next program address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.3 ADCNST (Add Constant)

Syntax

```
ADCNST {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[control={OFF | ON}]
remote={label | 9-bit unsigned integer}
min_off={25-bit unsigned integer}
data={25-bit unsigned integer}
[hr_data={7-bit unsigned integer}]
}
```

Figure 17-73. ADCNST Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Reserved	BRK	Next program address	0101	Remote address
6	3	1	9	4	9

Figure 17-74. ADCNST Control Field (C31:C0)

31–27	26–25	Res.	24–0
Reserved	Control	Res.	Minimum offset
5	1	1	25

Figure 17-75. ADCNST Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles Two

Register modified Register T (implicity)

Description ADCNST is an extension of ADM32. ADCNST first checks whether the data field value at the remote address is zero; it then performs different adds and moves on the result. ADCNST is typically used to extend the counter value of PWCNT.

min_off A 25-bit constant value that is added to the data field value if the remote data field is null.

data A 25-bit value that is always added to the remote data field.
Default: 0.

hr_data Seven least significant bits of the data addition to the remote data field.
Default: 0.

Figure 17-76 and Figure 17-77 illustrate the behavior of ADCNST if the remote data field is zero or is not zero.

Figure 17-76. ADCNST Operation If Remote Data Field[31:7] Is Not Zero

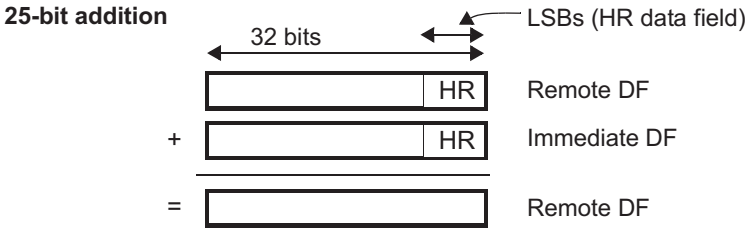
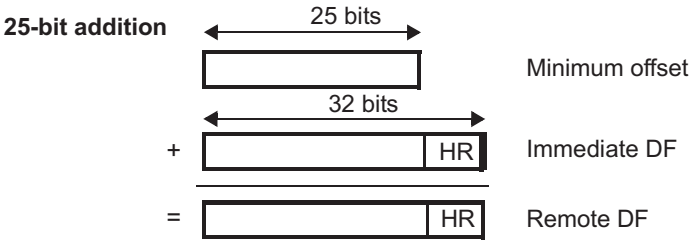


Figure 17-77. ADCNST Operation if Remote Data Field [31:7] Is Zero



Execution

```
If (Remote Data Field Value [31:7] != 0)
      Remote Data Field = Immediate Data Field + Remote Data Field;
else
      Remote Data Field = Immediate Data Field + min. offset(bits C24:C0);

Jump to Next Program Address;
```

17.5.3.4 ADC, ADD, AND, OR, SBB, SUB, XOR

Syntax

```
ADC | ADD | AND | OR | SBB | SUB | XOR {
src1 = { ZERO | IMM | A | B | R | S | T | ONES | REM | REMP }
src2 = { ZERO | IMM | A | B | R | S | T | ONES }
dest = { NONE | IMM | A | B | R | S | T }
[rdest = { NONE | REM | REMP }]
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[remote={label | 9-bit unsigned integer}]
[control={OFF | ON}]
[init={OFF | ON}]
[smode = {LSL | CSL | LSR | CSR | RR | CRR | ASR }]
[scount = {5 bit unsigned integer}]
[data={25-bit unsigned integer}]
[hr_data={7-bit unsigned integer}]
}
```

Figure 17-78. ADC, ADD, AND, OR, SBB, SUB, XOR Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Reserved	BRK	Next program address	0100	Remote address
6	3	1	9	4	9

Figure 17-79. ADC, ADD, AND, OR, SBB, SUB, XOR Control Field (C31:C0)

31–27	26	25–23	22–19	18–16
Reserved	Control	Sub Opcode	Src1	Src2
5	1	3	4	3

15–13	12–8	7	6	5	4–3	2–1	0
Smode	Scount	Ext. Reg	Init flag	1	Rdest	Register select	Res.
3	5	1	1	1	2	2	1

Figure 17-80. ADC, ADD, AND, OR, SBB, SUB, XOR Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles One to three cycles, depending on operands selected. (See Table 17-55)

Register modified Selected register (A, B, R, S, T, or NONE)

Description This instruction performs the specified 32-bit arithmetic or logical operation on operands src1 and src2, followed by an optional shift/rotate step. The result of this operation is then stored to either an N2HET register or the immediate data field of the instruction. In addition, the same result may be stored in a remote data field or the least signficant bits of a remote instruction program field (P[8:0]). Bits P[8:0] of the program field are used by most instructions formats to hold the remote address that the instruction operates on, so the ability to update this field programatically makes it easier to write subroutines that operate on different data sets.

The Sub-Opcode field C[25:3] determines which type of operation (ADD, ADC, AND, OR, SBB, SUB, XOR) is executed by the instruction. A list of these operations and the corresponding Sub-Opcode encoding can be found in Table 17-51.

All arithmetic is performed using 32-bit integer math. However, source and destination operands vary in width and can be 9 bits (REMP), 25 bits (A, B) or 32 bits (R,S,T, IMM, REM). Source operands REMP, A,B are extended to 32-bits before being operated on. Also the result of the computation needs to be truncated before being written back to REMP, A, or B when these are selected as destination operands. Table 17-52 provides a list of source operand options, how they are expanded to 32-bit integers (if applicable) and the control field encoding to select the option for src1 and src2 operands.

Table 17-53 provides a similar list of destination operands and their encodings. Up to two destination operands may be selected for each instruction, a register/immediate destination and a remote destination may be selected simultaneously. Truncation is performed independently for each destination operand as appropriate to its size.

An optional shift step following the arithmetic or logical operation may be selected through the smode and scount operands. The shift or rotate type is selected by the smode field; Table 17-54 illustrates the options that are available for smode. The number of bits shifted is determined by the scount operand.

Table 17-51. Arithmetic / Bitwise Logic Sub-Opcodes

Instruction	Description	Operation	Sub-Opcode
ADC	Add with Carry	result = src1 + src2 + C	C[25:23] = 011
ADD	Add	result = src1 + src2	C[25:23] = 001
AND	Bitwise Logic And	result = src1 & src2	C[25:23] = 010
OR	Bitwise Logic Or	result = src1 | src2	C[25:23] = 100
SBB	Subtract with Borrow	result = src1 - src2 - C	C[25:23] = 110
SUB	Subtract	result = src1 - src2	C[25:23] = 101
XOR	Bitwise Logic Exclusive Or	result = src1 ^ src2	C[25:23] = 111

Table 17-52. Source Operand Choices

Source Operand	32-bit value	Address	src1	src2
A	{A[24:0], 0x00}	n/a	C[22:19] = 0010	C[18:16] = 010
B	{B[24:0], 0x00}	n/a	C[22:19] = 0011	C[18:16] = 011
R	R[31:0]	n/a	C[22:19] = 0100	C[18:16] = 100
S	S[31:0]	n/a	C[22:19] = 0101	C[18:16] = 101
T	T[31:0]	n/a	C[22:19] = 0110	C[18:16] = 110
IMM	D[31:0]	current instruction address	C[22:19] = 0001	C[18:16] = 001
ZERO	0x00000000	n/a	C[22:19] = 0000	C[18:16] = 000
ONES	0xFFFFFFFF	n/a	C[22:19] = 0111	C[18:16] = 111
REM	D[31:0]	specified by remote[8:0]	C[22:19] = 1000	n/a
REMP	{0x000000, P[8:0]}	specified by remote[8:0]	C[22:19] = 1001	n/a

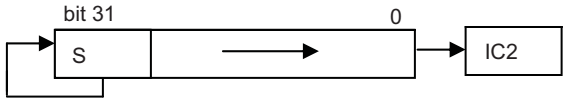
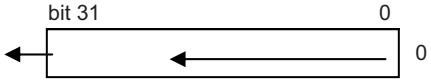
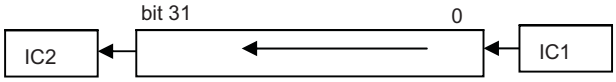
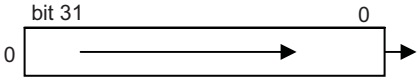
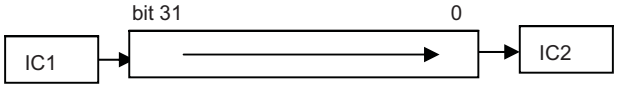
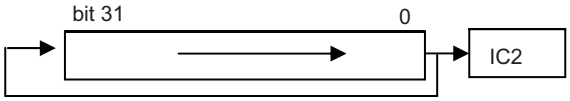
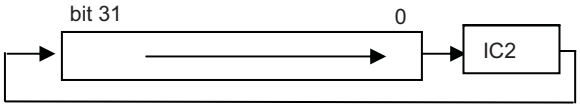
Table 17-53. Destination Operand Choices

Destination Operand	Stored Value	Address	dest	rdest
A	A[24:0] = result [31:8]	n/a	C[7] = 0, C[2:1] = 00	n/a
B	B[24:0] = result [31:8]	n/a	C[7] = 0, C[2:1] = 01	n/a
R	R[24:0] = result [31:0]	n/a	C[7] = 1, C[2:1] = 00	n/a
S	S[24:0] = result [31:0]	n/a	C[7] = 1, C[2:1] = 01	n/a
T	T[24:0] = result [31:0]	n/a	C[7] = 0, C[2:1] = 10	n/a

Table 17-53. Destination Operand Choices (continued)

Destination Operand	Stored Value	Address	dest	rdest
IMM	D[31:0] = result [31:0]	current instruction address	C[7] = 1, C[2:1] = 10	n/a
NONE	n/a	n/a	C[7] = 0, C[2:1] = 11	C[4:3] = 00
REM	D[31:0] = result [31:0]	specified by remote[8:0]	n/a	C[4:3] = 01
REMP	P[8:0] = result [8:0]	specified by remote[8:0]	n/a	C[4:3] = 10

Table 17-54. Shift Encoding

Shift Type	C[15:13] smode	Operation Illustrated (1)
No Shift Applied	0 0 0	n/a - no shift
ASR-Arithmetic Shift Right	0 0 1	
LSL-Logical Shift Left	0 1 0	
CSL-Carry Shift Left	0 1 1	
LSR-Logical Shift Right	1 0 0	
CSR-Carry Shift Right	1 0 1	
RR - Rotate Right	1 1 0	
CRR – Carry Rotate Right	1 1 1	

(1) IC1 is the carry flag after the arithmetic / logical operation is performed. Ic2 is the updated carry flag after the shift operation is performed. s is the sign bit.

Table 17-55. Execution Time for ADC, ADD, AND, OR, SBB, SUB, XOR Instructions

src1	dest	rdest	remote[8:0]	Cycles
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	NONE	! = next[8:0]	1
REM or REMP	A,B,R,S,T, or NONE	NONE	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	IMM	REM	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	REMP	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	NONE	== next[8:0]	2
REM or REMP	IMM	REM	x	3
x	IMM	REMP	x	3
REM or REMP	x	REM	== next[8:0]	3
x	x	REMP	== next[8:0]	3

Execution

```
         / Notes: IR1, IR2 are 32-bit intermediate results
   // SRC1, SRC2 are 32-bit sources selected
   //            by fields src1, src2
   // IC1, IC2 are intermediate values of the carry flag
   // IZ1, IZ2 are intermediate values of the zero flag
   // IN1, IN2 are intermediate values of the negative flag
   // IV1, IV2 are intermediate values of the overflow flag
   // scount is the shift count (0 to 31) specified by C12:C8

   /********** SOURCE OPERAND DECODING STAGE **********/
   switch (C22:C19)
{
     case 0000:SRC1[31:0] = 0x00000000
     case 0001:SRC1[31:0] = Immediate Data Field D[31:0]
     case 0010:SRC1[31:8] = A[24:0]; SRC1[6:0] = 0
     case 0011:SRC1[31:8] = B[24:0]; SRC1[6:0] = 0
     case 0100:SRC1[31:0] = R[31:0]
     case 0101:SRC1[31:0] = S[31:0]
     case 0110:SRC1[31:0] = T[31:0]
     case 0111:SRC1[31:0] = 0xFFFFFFFF
     case 1000:SRC1[31:0] = Remote Data Field D[31:0]
     case 1001:SRC1[31:9] = 0; SRC1[8:0] = Remote Program Field P[8:0]
   }

switch (C18:C16)
{
     case 000:SRC2[31:0] = 0x00000000
     case 001:SRC2[31:0] = Immediate Data Field[31:0]
     case 010:SRC2[31:8] = A[24:0]; SRC2[6:0] = 0
     case 011:SRC2[31:8] = B[24:0]; SRC2[6:0] = 0
     case 100:SRC2[31:0] = R[31:0]
     case 101:SRC2[31:0] = S[31:0]
     case 110:SRC2[31:0] = T[31:0]
     case 111:SRC2[31:0] = 0xFFFFFFFF
   }
   /******** ARITHMETIC / LOGICAL OPERATION STAGE *******/
   switch (C[25:23])
   {
     case 011:IR1 = src1 + src2 + C // ADC
     case 001:IR1 = src1 + src2 // ADD
     case 010:IR1 = src1 & src2 // AND
     case 100:IR1 = src1 | src2 // OR
     case 110:IR1 = src1 - src2 - C // SBB
     case 101:IR1 = src1 - src2 // SUB
     case 111:IR1 = src1 ^ src2 // XOR
   }

   IC1 = Carry Out if Operation is ADD, ADC, SUB, SBB
         = 0 if Operation is AND, OR, XOR
   IZ1 = Set if IR1 is zero, Clear if IR1 is non-zero
   IN1 = IR[31]
   IV1 = (IC1 XOR IR1[31]) AND NOT(SRC1[31] XOR SRC2[31])

   /******************* SHIFT STAGE ******************/
   switch (C15:C13)
{
     case 000: // smode = No Shift
           IR2 = IR1
           IC2 = IC1; IZ2 = IZ1; IN2 = IN1; IV2 = IV1;

     case 001: // smode = Arithmetic Shift Right
           IR2[31 - scount : 0] = IR1[31:scount]

           if (scount>0) {
              IR2[31 : 31 - scount + 1] = IR1[31]
```

```
            IC2 = IR1[scount-1]
         }
         else {
            IC2 = IC1
         }

         IN2 = IR2[31];
         if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
         IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 010: // smode = Logical Shift Left
         IR2[31 : scount] = IR1[31 - scount: 0]

         if (scount > 0) {
            IR2[scount - 1 : 0] = 0
         }

         IC2 = IC1
         IN2 = IR2[31];
         if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
         IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 011: // smode = Carry Shift Left
         IR2[31 : scount] = IR1[31 - scount: 0]

   if (scount>0) {
            IR2[scount - 1 : 0] = [IC1,...IC1]
            IC2 = IR1[31 - scount + 1]
         }
         else
   {
            IC2 = IC1
         }

   IN2 = IR2[31];
         if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
         IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 100: // smode = Logical Shift Right
         IR2[31 - scount : 0] = IR1[31:scount]

         if (scount>0) {
            IR2[31 : 31 - scount + 1] = 0
         }

         IC2 = IC1
         IN2 = IR2[31];
         if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
         IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 101: // smode = Carry Shift Right
         IR2[31 - scount : 0] = IR1[31:scount]

         if(scount>0) {
            IR2[31:31-scount + 1] = [IC1,...IC1]
            IC2 = IR1[scount-1]
         }
         else {
            IC2 = IC1
         }

         IN2 = IR2[31];
         IZ2 = Set if IR2 == 0;
         IV2 = (IR2[31] XOR IR1[31]) OR IV1

case 110: // smode = Rotate Right
```

```
            IR2[31 - scount : 0] = IR1[31:scount]

            if(scount>0) {
               IR2[31:31-scount+1] = IR1[scount-1:0]
               IC2 = IR1[scount-1]
            }
            else {
               IC2 = IC1
            }

            IN2 = IR2[31];
            if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
            IV2 = (IR2[31] XOR IR1[31]) OR IV1

      case 111: // smode = Carry Rotate Right
            IR2[31 - scount : 0] = IR1[31:scount]

            if (scount == 0) {
               IC2 = IC1
            }
            else if (scount == 1) {
               IR2[31] = IC1
               IC2 = IR1[0]
            }
            else {
               IR2[31:31-scount+1] = {IR1[scount-2:0],IC1}
               IC2 = IR1[scount - 1]
            }

            IN2 = IR2[31];
            if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
            IV2 = (IR2[31] XOR IR1[31]) OR IV1
      }
      /********** WRITE REGISTER DESTINATION STAGE ***********/
      switch (C7, C2:C1)
      {
            case 000:A[24:0] = IR2[31:8]
            case 001:B[24:0] = IR2[31:8]
            case 010:T[31:0] = IR2[31:0]
            case 011:IR2 is not stored in register, immediate
            case 100:R[31:0] = IR2[31:0]
            case 101:S[31:0] = IR2[31:0]
            case 110:Immediate Data Field[31:0] = IR2
            case 111:IR2 is not stored in register, immediate
      }
      /*********** WRITE REMOTE DESTINATION STAGE ***********/
      switch (C4:3)
   {
            case 00:IR2 is not stored in remote field
            case 01:Remote Data Field D[31:0] = IR2
            case 10:Remote Program Field P[8:0] = IR2[8:0]
            case 11:IR2 is not stored in remote field
      }
      /***************** UPDATE FLAGS STAGE *****************/
      C FLAG = IC2
      N FLAG = IN2
      Z FLAG = IZ2
      V FLAG = IV2
      If (Init Flag == 1)
      {
            ACF = 0;
            DCF = 1;
            GPF = 0;
            NAF = 0;
      }
      else ACF, DCF, GPF, NAF remain unchanged;
```

17.5.3.5 ADM32 (Add Move 32)

Syntax

```
ADM32 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[control={OFF | ON}]
[init={OFF | ON}]
type={IM&REGTOREG | REM&REGTOREG | IM&REMTOREG |
IM&REGTOREM}
reg={A | B | R | S | T }
data={25-bit unsigned integer}
[hr_data={7-bit unsigned integer}]
}
```

Figure 17-81. ADM32 Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Reserved	BRK	Next program address	0100	Remote address
6	3	1	9	4	9

Figure 17-82. ADM32 Control Field (C31:C0)

31–27	26	25–23	22–16
Reserved	Control	000	Reserved
5	1	3	15

15–8	7	6	5	4–3	2–1	0
Reserved	Ext Reg	Init flag	1	Move type	Register select	Res.
15	1	1	1	2	2	1

Figure 17-83. ADM32 Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles One or two cycles (see Table 17-56)

Register modified Selected register (A, B, R, S, or T)

Description This instruction modifies the selected ALU register or data field values at the remote address depending on the move type. The modified value results from adding the immediate or remote data field to the ALU register or the remote data field, depending on the move type. Table description shows the C2 and C1 bit encoding for determining which register is selected.

init (Optional) Determines whether or not system flags are initialized. A value of ON reinitializes the following system flags to these states:

Acceleration flag (ACF) = 0

Deceleration flag (DCF) = 1

Gap flag (GPF) = 0

New angle flag (NAF) = 0
A value of OFF results in no change to the system flags.
Default: OFF

type Specifies the move type to be executed.

Table 17-56. Move Types for ADM32

Type	C4	C3	Add	Destination(s)	Cycles
IM&REGTOREG	0	0	Imm. data field + Reg. A, B, R, S, or T	Register A, B, R, S, or T	1
REM&REGTOREG	0	1	Remote data field + Reg. A, B, R, S, or T	Register A, B, R, S, or T	2
IM&REMTOREG	1	0	Imm. data field + Remote data field	Register A, B, R, S, or T	2
IM&REGTOREM	1	1	Imm. data field + Reg. A, B, R, S, or T	Remote data field	1

If selected register is R, S, or T, the operation is a 32-bit Addition/move. If A or B register is selected, it is limited to 25-bit operation since A and B only support 25-bit.

data Specifies the 25-bit integer value for the immediate data field.

hr_data Specifies the 7 least significant bits of the immediate data field.
Default: 0.

Execution

```
switch (C4:C3)
{
      case 00:
           Selected register = Selected register + Immediate Data Field;
      case 01:
           Selected register = Selected register + Remote Data Field;
      case 10:
           Selected register = Immediate Data Field + Remote Data Field;
      case 11:
           Remote Data Field = Selected register + Immediate Data Field;
}

If (Init Flag == 1)
{
      ACF = 0;
      DCF = 1;
      GPF = 0;
      NAF = 0;
}
else
      All flags remain unchanged;

Jump to Next Program Address;
```

Figure 17-84 and Figure 17-85 illustrate the ADM32 operation for various cases.

Figure 17-84. ADM32 Add and Move Operation for IM&REGTOREG (Case 00)

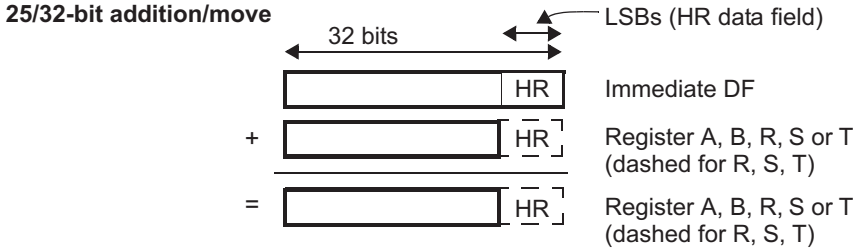
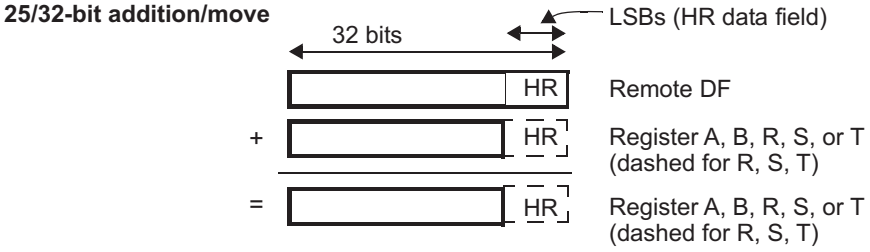


Figure 17-85. ADM32 Add and Move Operation for REM&REGTOREG (Case 01)



17.5.3.6 APCNT (Angle Period Count)

Syntax

APCNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
[irq={OFF | ON}]
type={FALL2FALL | RISE2RISE}
[control={OFF | ON}]
prv={OFF | ON}}]
period={25-bit unsigned integer}
data={25-bit unsigned integer}
}

Figure 17-86. APCNT Program Field (P31:P0)

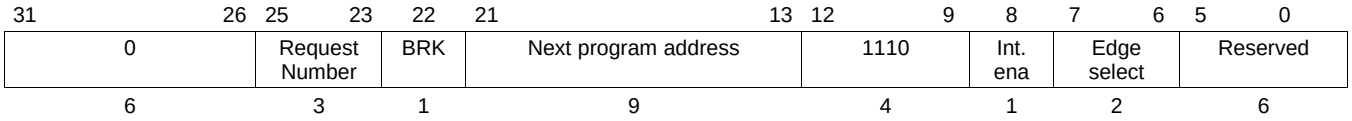


Figure 17-87. APCNT Control Field (C31:C0)

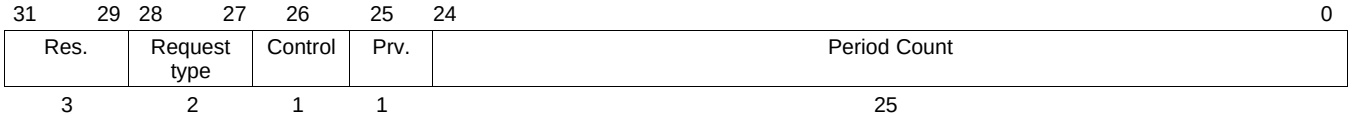
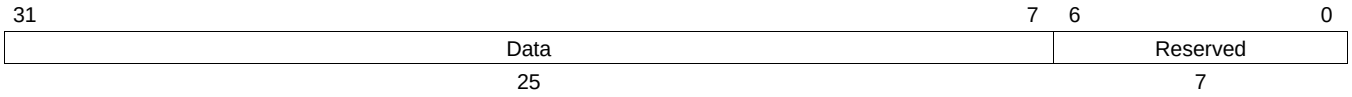


Figure 17-88. APCNT Data Field (D31:D0)



Cycles

One or two cycles

- Cycle 1: edge detected (normal operation)
- Cycle 2: edge detected and GPF = 1 and underflow condition is true

One cycle (normal operation) two cycles (edge detected)

Register modified

Register A and T (implicitly)

Description

This instruction is used before SCNT and ACNT to generate an angle-referenced time base synchronized to an external signal (that is, a toothed wheel signal). It is assumed that the pin and edge selections are the same for APCNT and ACNT.

APCNT is restricted to pin HET[2]. The toothed wheel must then be connected to pin HET[2].

APCNT uses the gap flag (GPF) defined by ACNT to start or stop captures in the period count field [C24:C0]. When GPF = 1, the previous period value is held in the control field and in register T. When GPF = 0, the current period value is captured in the control field and in register T.

APCNT uses the step width flags (SWF0 and SWF1) defined by SCNT to detect period durations shorter than one step, and then disables capture.

The edge select encoding is shown in Table 17-57.

irq ON generates an interrupt when the edge state is satisfied. OFF prevents an interrupt from being generated.
Default: OFF.

type Specifies the edge type that triggers the instruction.
Default: Fall2Fall.

Table 17-57. Edge Select Encoding for APCNT

type	P7	P6	Selected Condition
Fall2Fall	1	0	Falling edge
Rise2Rise	1	1	Rising edge

period Contains the 25-bit count value from the previous APCNT period.

data 25-bit value serving as a counter.
Default: 0.

Execution

```
Z = 0;

If (Data field register != 1FFFFFFh)
{
   Register A = Data field register + 1;
   Data field register = Data field register + 1;
}
elseIf (specified edge not detected on HET[2])
{
   Register A = 1FFFFFFh;
   APCNT Ovflw flag = 1;
}

If (specified edge detected on HET[2])
{
   Z = 1;

   If (Data field register == 1FFFFFFh)
   {
      Register A = 1FFFFFFh;
      Register T = 1FFFFFFh;Period count = 1FFFFFFh;
      Period count = 1FFFFFFh;
   }
   elseIf (GPF == 0 AND Data Field register >= Step width)
   {
      Register A = Data field register + 1;
      Register T = Register A;
      Period count = Register T;

      If (Interrupt Enable == 1)
         HETFLG[n] = 1;      /* n depends on address */
      If ([C28:C27] == 01)
         Generate request on request line [P25:P23];
      If ([C28:C27] == 11)
         Generate quiet request on request line [P25:P23];
   }

   If (GPF == 1)
         Register T = Period count;
   If (Data Field register < Step width)
   {
      Register T = Period count;
      APCNT Undflw flag = 1;
      Period Count = 000000h;
   }

   Data field register = 000000h;
}
else
{
      Register T = Period count;
}

Prv bit = Current Lx value of HET[2] pin;

Jump to Next Program Address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.7 BR (Branch)

Syntax

BR {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[prv={OFF | ON}]
cond_addr={label | 9-bit unsigned integer}
[pin= {pin number}]
event={NOCOND | FALL | RISE | BOTH | ZERO | NAF | LOW | HIGH | C | NC | EQ | Z | NE | NZ | N | PZ | V | NV | ZN | P | GE | LT | GT | LE | LO | HS }
[irq={OFF | ON}]
}

Figure 17-89. BR Program Field (P31:P0)

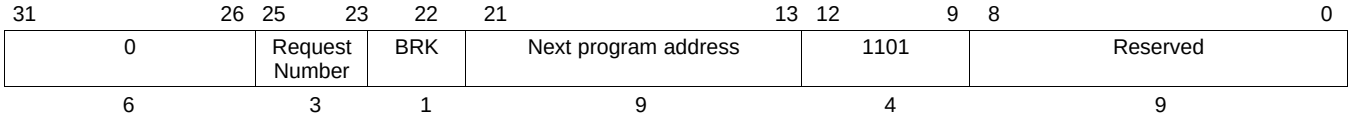


Figure 17-90. BR Control Field (C31:C0)

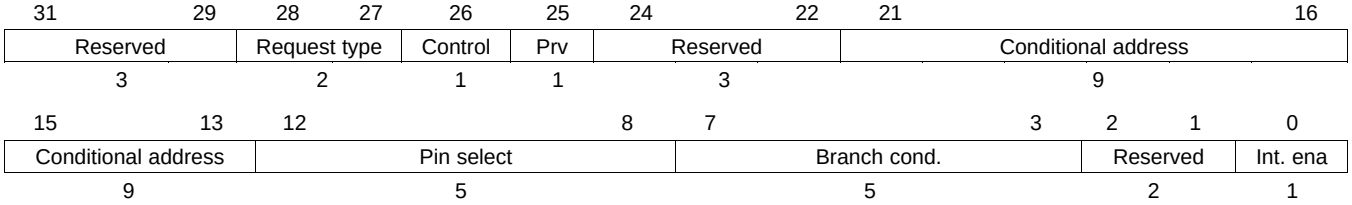
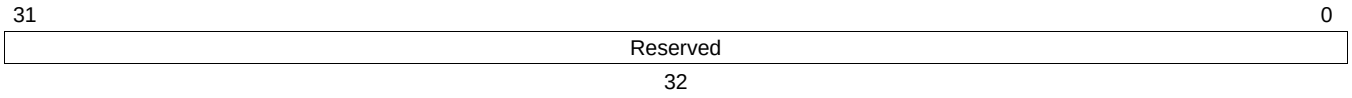


Figure 17-91. BR Data Field (D31:D0)



Cycles One

Register modified None

Description This instruction executes a jump to the conditional address [C21:C13] on a pin or a flag condition, and can be used with all pins.

Table 17-58 provides the branch condition encoding.

event Specifies the event that triggers a jump to the indexed program address.

Default: FALL

irq ON generates an interrupt when the event occurs that triggers the jump. If irq is set to OFF, no interrupt is generated.
Default: OFF.

Table 17-58. Branch Condition Encoding for BR

Event	C7	C6	C5	C4	C3	Branch Condition
NOCOND	0	0	0	0	0	Always
FALL	0	0	1	0	0	On falling edge on the selected pin
RISE	0	1	0	0	0	On rising edge on selected pin
BOTH	0	1	1	0	0	On rising or falling edge on selected pin
ZERO	1	0	0	0	0	If Zero flag is set
NAF	1	0	1	0	0	If NAF_global flag is set
LOW	1	1	0	0	0	On LOW level on selected pin
HIGH	1	1	1	0	0	On HIGH level on selected pin
C	0	0	0	0	1	Carry Set: C==1
NC	0	0	0	1	1	Carry Not Set: C==0
EQ, Z	0	0	1	0	1	Equal or Zero: Z==1
NE, NZ	0	0	1	1	1	Not Equal or Not Zero: Z==0
N	0	1	0	0	1	Negative: N==1
PZ	0	0	1	1	1	Positive or Zero: N==0
V	0	1	1	0	1	Overflow: V==1
NV	0	1	1	1	1	No Overflow: V==0
ZN	1	0	0	0	1	Zero or Negative: (Z OR N) == 1
P	1	0	0	1	1	Positive: (Z OR N) == 0
GE	1	0	1	1	1	Signed Greater Than or Equal: (N XOR V) == 0
L	1	0	1	0	1	Signed Less Than (N XOR V) == 1
G	1	1	0	1	1	Signed Greater Than (Z OR (N XOR V)) == 0
LE	1	1	0	0	1	Signed Less Than (Z OR (N XOR V)) == 1
LO	1	1	1	1	1	Unsigned Less Than: (C OR Z) == 0
HS	1	1	1	0	1	Unsigned Higher or Same (C OR Z) == 1

Execution

```
If (Condition is true)
{
   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on request line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
   Jump to Conditional Address;
}
else
{
   Jump to Next Program Address;
}

Prv bit = Current Lx value of selected pin; (Always Executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.8 CNT (Count)

Syntax

```
CNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
[angle_count={OFF | ON}]
[reg={A | B | T | NONE}]
[comp ={EQ | GE}]
[irq={OFF | ON}]
[control={OFF | ON}]
max={25-bit unsigned integer}
[data={25-bit unsigned integer]
}
```

Figure 17-92. CNT Program Field (P31:P0)

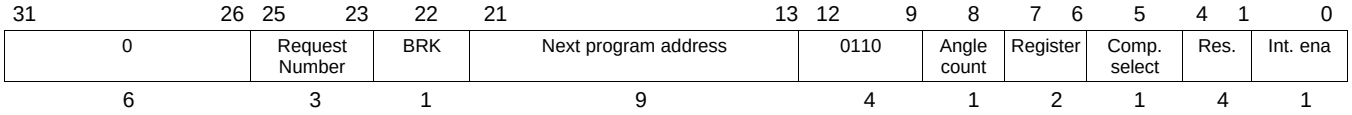


Figure 17-93. CNT Control Field (C31:C0)

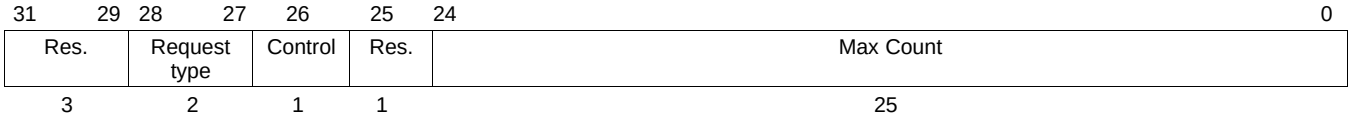
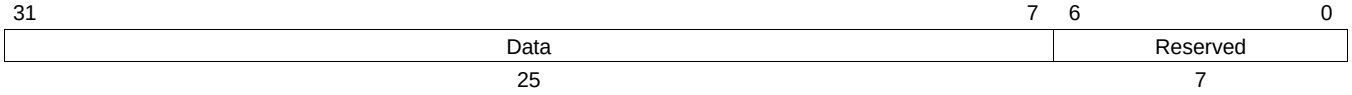


Figure 17-94. CNT Data Field (D31:D0)



Cycles

One or two

One cycle (time mode), two cycles (angle mode)

Register modified

Selected register (A, B or T)

Description

This instruction defines a virtual timer. The counter value stored in the data field [D31:7] is incremented unconditionally on each execution of the instruction when in time mode (angle count bit [P8] = 0). When the count reaches the maximum count specified in the control field, the counter is reset. It takes one cycle in this mode.

In angle mode (angle count bit [P8] = 1), CNT needs data from the software angle generator (SWAG). When in angle count mode the angle increment value will be 0 or 1. It takes two cycles in this mode.

angle_count	Specifies when the counter is incremented. A value of ON causes the counter value to be incremented only if the new angle flag is set (NAF_global = 1). A value of OFF increments the counter each time the CNT instruction is executed. Default value for this field is OFF.
comp	When set to EQ the counter is reset, when it is equal to the maximum count. When set to GE the counter is reset, when it is greater or equal to the maximum count. Default: GE.
irq	ON generates an interrupt when the counter overflows to zero. The interrupt is not generated until the data field is reset to zero. If irq is set to OFF, no interrupt is generated. Default: OFF.
max	Specifies the 25-bit integer value that defines the maximum count value allowed in the data field. When the count in the data field is equal to max, the data field is reset to 0 and the Z system flag is set to 1.
data	Specifies the 25-bit integer value serving as a counter. Default: 0.

Execution

```
Z = 0;

If (Angle Count (bit P8 == 1))
{
   If (NAF_global == 0)
   {
      Selected register = immediate data field;
      Jump to Next Program Address;
   }
   else
   {
      If ((Immediate Data Field + Angle Increment) >= Max count)
      {
         Z = 1;
         Selected register = ((Immediate Data Field + Angle Inc.) - Max count);
         Immediate Data Field = ((Immediate Data Field + Angle Inc.) - Max count);

         If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
         If ([C28:C27] == 01) Generate request on request line [P25:P23];
         If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
      }
      else
      {
         Selected register = Immediate Data Field + Angle Increment;
         Immediate Data Field = Immediate Data Field + Angle Increment;
      }
   }
}

else if(Time mode (bit P8 == 0))
{
   If [(P5==0) AND (Immediate Data Field == Max count)]
   OR [(P5==1) AND (Immediate Data Field >= Max count)]
   {
       Z = 1;
       Selected register = 00000;
       Immediate Data Field = 00000;

       If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
       If ([C28:C27] == 01) Generate request on request line [P25:P23];
       If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
   }
   else
   {
       Selected register = Immediate Data Field + 1;
       Immediate Data Field = Immediate Data Field + 1;
   }
}

Jump to Next Program Address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.9 DADM64 (Data Add Move 64)

Syntax

```
DADM64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
[pin={pin number}]
comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}]
}
```

-or-

Syntax

```
DADM64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
cntl_val={29-bit unsigned integer}
data={25-bit unsigned integer}
[hr_data= {7-bit unsigned integer}]
}
```

Figure 17-95. DADM64 Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Reserved	BRK	Next program address	0010	Remote Address
6	3	1	9	4	9

Figure 17-96. DADM64 Control Field (C31:C0)

31–29	28–27	26	25–23	22	21–16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15–13	12–8	7	6–5	4–3	2–1	0
Conditional address	Pin select	Ext Reg	Comp. mode	Action	Register select	Int. ena
9	5	1	2	2	2	1

Figure 17-97. DADM64 Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles Two

Register modified Register T (implicitly)

Description This instruction modifies the data field and the control field at the remote address. The remote data field value is not just replaced, but is added with the DADM64 data field.

DADM64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similarly to the format of the DADM64 control field. A second syntax, in which the entire 29-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar to the DADM64 control field. Either syntax may be used, but you must use one or the other but not a combination of syntaxes.

Figure 17-98 shows the DADM64 add and move operation.

Figure 17-98. DADM64 Add and Move Operation

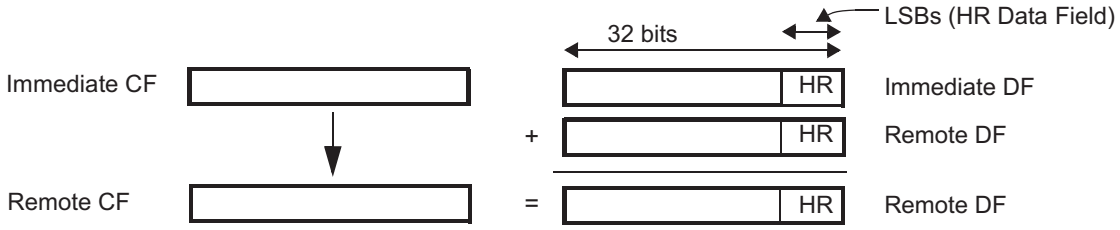


Table 17-59. DADM64 Control Field Description

request	maintains the control field for the remote instruction
control	maintains the control field for the remote instruction
en_pin_action	maintains the control field for the remote instruction
cond_addr	maintains the control field for the remote instruction

Table 17-59. DADM64 Control Field Description (continued)

pin	maintains the control field for the remote instruction
register	maintains the control field for the remote instruction
action	maintains the control field for the remote instruction
irq	maintains the control field for the remote instruction
data	Specifies the 25-bit initial value for the data field.
hr_data	Seven least significant bits of the 32 bit data field. Default: 0
cntl_val	Specifies the 29 least significant bits of the Control field.

Execution

```
Remote Data Field = Remote Data Field + Immediate Data Field;
Register T = Immediate Data Field;
Remote Control Field = Immediate Control Field;
Jump to Next Program Address;
```

17.5.3.10 DJZ (Decrement and Jump if Zero)

NOTE: DJNZ is also a supported syntax. The functionality of the two instruction names is identical.

Syntax

```
DJZ {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
}
```

Figure 17-99. DJZ Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8	7–6	5–0
0	Request Number	BRK	Next program address	1010	Res.	10	Reserved
6	3	1	9	4	1	2	6

Figure 17-100. DJZ Control Field (C31:C0)

31–29	28–27	26	25–22	21–16
Reserved	Request type	Control	Reserved	Conditional address
3	2	1	4	9

15–13	12–8	7–3	2–1	0
Conditional address	Reserved		Register select	Int. ena
9	10		2	1

Figure 17-101. DJZ Data Field (D31:D0)

31–7	6–0
Data	Reserved
25	7

Cycles One

Register modified Selected register (A, B, or T)

Description This instruction defines a virtual down counter used for delayed execution of certain instructions (to generate minimum on/off times). When DJZ is executed with counter value not zero, the counter value is decremented. If the counter value is zero, the counter remains zero until it is reloaded with a non-zero value. The program flow can be modified when down counter value is zero by using the conditional address.

cond_addr	This field is not optional for the DJZ instruction.
irq	ON generates an interrupt when the data field reaches zero. No interrupt is generated when the bit is OFF. Default: OFF.
data	Specifies the 25-bit integer value used as a counter. This counter is decremented each time the DJZ instruction is executed until the counter reaches 0. Default: 0.

Execution

```
If (Data != 0)
{
   Data = Selected register = Data - 1;
   Jump to Next Program Address;
}
else
{
   Selected register = 000000h;

   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on request line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

   Jump to conditional Address;
}
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.11 ECMP (Equality Compare)

Syntax

```
ECMP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[hr_lr={HIGH | LOW}]
[angle_comp={OFF | ON}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin={pin number}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | R | S| T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 17-102. ECMP Program Field (P31:P0)

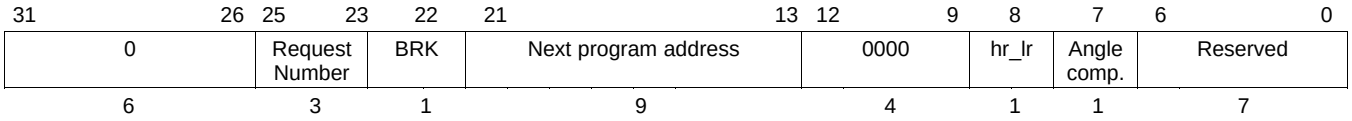


Figure 17-103. ECMP Control Field (C31:C0)

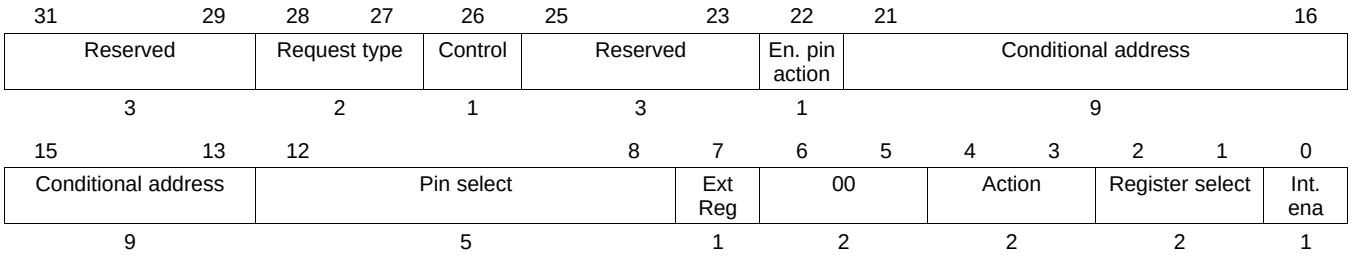
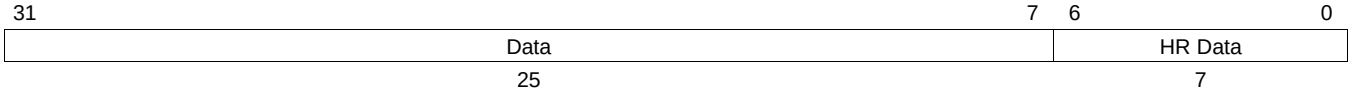


Figure 17-104. ECMP Data Field (D31:D0)



Cycles	One
Register modified	Register A, B, R, S or T if selected
Description	ECMP can use all pins. This instruction compares a 25-bit data value stored in the data field (D31–D7) to the value stored in the selected ALU register (A, B, R, S, or T). Register select encoding can be found in Section 17.5.2. If R, S, or T registers are selected, and if the 25-bit data field matches, ECMP updates the register with the 32-bit value (D31-D0). If the hr_lr bit is cleared, the pin action will occur after a high resolution delay from the next loop resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in the data field (D6–D0). The behavior of the pins is governed by the four action options in bits C4:C3. ECMP uses the zero flag to generate opposite pin action (synchronized to the loop resolution clock).

angle_comp	Determines if an angle compare is performed. A value of ON causes the comparison to be performed only if the new angle flag is set (NAF = 1). If OFF is specified, the compare is then performed regardless of the state of the new angle flag. Default: OFF.
irq	Specifies whether or not an interrupt is generated. A value of ON sends an interrupt if register and data field values are equivalent. If OFF is selected, no interrupt is generated. Default: OFF.
data	Specifies the value for the data field. This value is compared with the selected register.
hr_data	Specifies the HR delay. Default: 0.

Execution

```
If (Angle Comp. bit == 0 OR (Angle Comp. bit == 1 AND NAF_global == 1))
{

   If (Selected register value == Immediate data field value)
   {

       If (hr_lr bit == 0)
       {
          If (Enable Pin action == 1)
          {
             Selected Pin = Pin Action AT next loop resolution clock + HR delay;
          }
       }
       else
       {
          If (Enable Pin action == 1)
          {
             Selected Pin = Pin Action AT next loop resolution clock;
          }
       }

       If (Z == 1 AND Opposite action == 1)
       {
          If (Enable Pin action == 1)
          {
             Selected Pin = opposite Pin Action AT next loop resolution clock;
          }

          If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
          If ([C28:C27] == 01) Generate request on request line [P25:P23];
          If ([C28:C27] == 11)Generate quiet request on request line [P25:P23];

          If (register R is selected) R register = Compare value (32 bit);
          If (register S is selected) S register = Compare value (32 bit);
          If (register T is selected) T register = Compare value (32 bit);

          Jump to Conditional Address;
       }
   }
   elseIf (Z == 1 AND Opposite action == 1)
   {
      If (Enable Pin action == 1)
      {
         Selected Pin = opposite Pin Action AT next loop resolution clock;
      }
      Jump to Next Program Address;
}
else  // Angle Comp. bit == 1 AND NAF_global == 0
{
   Jump to Next Program Address;
}
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.12 ECNT (Event Count)

Syntax

```
ECNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin={pin number}
event={NAF | FALL | RISE | BOTH | ACCUHIGH | ACCULOW}
[reg={A | B | R| S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
}
```

Figure 17-105. ECNT Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8	7–6	5–0
0	Request Number	BRK	Next program address	1010	Res.	01	Reserved
6	3	1	9	4	1	2	6

Figure 17-106. ECNT Control Field (C31:C0)

31–29	28–27	26	25	24–22	21–16
Reserved	Request type	Control	Prv.	Reserved	Conditional address
3	2	1	1	3	9

15–13	12–8	7	6–4	3	2–1	0
Conditional address	Pin select	Ext Reg	Event	Res.	Register select	Int. ena
9	5	1	3	1	2	1

Figure 17-107. ECNT Data Field (D31:D0)

31–7	6–0
Data	Reserved
25	7

Cycles One cycle

Register modified Selected Register (A, B, R, S, T or none)

Description This instruction defines a specialized 25-bit virtual counter used as an event counter or pulse accumulator (see Table 17-60). The counter value is stored in the data field [D31:D7] and the selected register. If one of the 32-bit registers (R,S,T) is selected, the 25 bit count value is stored left justified in the register with zeros in the seven least significant bits.

When an event count condition is specified, the counter value is incremented on a pin edge condition or on the NAF condition (NAF is defined in ACNT). This instruction can be used with all pins.

event The event that triggers the counter.

Table 17-60. Event Encoding Format for ECNT

Event	C6	C5	C4	Count Conditions	Mode	Int. Available
NAF	0	0	0	NAF flag is Set	Angle counter	Y
FALL	0	0	1	Falling edge on selected pin	Event counter	Y
RISE	0	1	0	Rising edge on selected pin	Event counter	Y
BOTH	0	1	1	Rising and Falling edge on selected pin	Event counter	Y
ACCUHIGH	1	0	-	while pin is high level	Pulse accumulation	N
ACCULOW	1	1	-	while pin is low level	Pulse accumulation	N

irq ON generates an interrupt when event in counter mode occurs. No interrupt is generated with OFF.

Default: OFF.

data 25-bit integer value serving as a counter.

Default: 0.

Execution

```
If (event occurs)
{
   If (Register A or B Selected) {
      Selected register = Immediate Data Field + 1;
   }

   If (Register R, S or T Selected)
   {
      Selected register[31:7] = Immediate Data Field + 1;
      Selected register[6:0] = 0;
   }

   Immediate Data Field = Immediate Data Field + 1;

   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on line [P25:P23];

   Jump to Conditional Address;
}
else
{
    Jump to Next Program Address;
}

Prv bit = Current Logic (Lx) value of selected pin; (Always executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.13 MCMP (Magnitude Compare)

Syntax

```
MCMP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[hr_lr={LOW |HIGH}]
[angle_comp={OFF | ON}]
[savesub={OFF | ON}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin={pin number}
order={REG_GE_DATA | DATA_GE_REG}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | R | S | T | NONE}
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 17-108. MCMP Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8	7	6	5	4–0
0	Request Number	BRK	Next program address	0000	hr_lr	Angle comp.	Res.	Save sub.	Res.
6	3	1	9	4	1	1	1	1	5

Figure 17-109. MCMP Control Field (C31:C0)

31–29	28–27	26	25–23	22	21–16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15–13	12–8	7	6	5	4–3	2–1	0
Conditional address	Pin select	Ext Reg	1	Order	Action	Register select	Int. ena
9	5	1	1	1	2	2	1

Figure 17-110. MCMP Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles One

Register modified T (if save sub bit P[5] is set)

Description

This instruction compares the magnitude of the 25-bit data value stored in the data field (D31-D7) and the 25-bit value stored in the selected ALU register (A, B, R, S, or T).

If the hr_lr bit is reset, pin action will occur after a delay from the next loop resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in the data field (D6-D0).

When the data value matches, an output pin can be set or reset according to the pin action bit (C[4]). The pin will not change states if the enable pin action bit (C[22]) is reset.

MCMP uses the zero flag set to generate opposite pin action (synchronized to the loop resolution clock). The save sub bit (P[5]) provides the option to save the result of a subtraction into register T.

NOTE: The Difference Between Compare Values

The difference between the two data values must not exceed (2^{24}) - 1.

angle_comp Determines whether or not an angle compare is performed. A value of ON causes the comparison to be performed only if the new angle flag is set (NAF = 1). If OFF is specified, the compare is then performed regardless of the state of the new angle flag.

Default: OFF.

savesub When set, the comparison result is saved into the T register (upper 25 bits).

Default: OFF.

order Specifies the order of the operands for the comparison.

Table 17-61. Magnitude Compare Order for MCMP

Order	C5	Description
REG_GE_DATA	0	Evaluates to true if the register value is greater than or equal to the data field value.
DATA_GE_REG	1	Evaluates to true if the data field value is greater than or equal to the register value.

irq Specifies whether or not an interrupt is generated. A value of ON sends an interrupt if the compare match occurs according to the order selected. If OFF is selected, no interrupt is generated.

data Specifies the value for the data field. This value is compared with the selected register.

hr_data HR delay. The default value for an unspecified bit is 0.

Execution

```
If (Angle Compare P[7] == 0 OR (P[7] == 1 AND NAF_global == 1))
{
    If(  (Order C[5] == 1) AND (Data[31:7]- Selected register[31:7]) >= 0))
      OR ( (Order C[5] == 0) AND Selected register[31:7] - Data[31:7]) >= 0))
    {
       If (Order C[5] == 1 AND Save subtract P[5] == 1)
       {
          Register T[31:7] = Data[31:7] - Selected register[31:7];
          Register T[6:0] = 0;
       }

       If (Order C[5] == 0 AND Save subtract P[5] == 1)
       {
          Register T[31:7] = Selected register[31:7] - Data[31:7];
          Register T[6:0] = 0;
       }

       If (Enable Pin Action C[22] == 1)
       {
          If (hr_lr P[8] = 0) {
             Schedule Action on Selected Pin C[12:8] at start of next loop
             + HR Delay D[6:0];
          }
          else
          {
             Schedule Pin Action on Selected Pin C[12:8] at start of next loop;
          }
       }

       If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
       If ([C28:C27] == 01) Generate request on request line [P25:P23];
       If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

       Jump to Conditional Address;
    }
    else if (Z == 1 AND Opposite Action C[3] == 1 )
    {
       If (Enable Pin Action C[22] == 1)
       {
          Schedule Opposite Pin Action on Selected Pin C[12:8] at start of next loop;
       }

       Jump to Next Program Address;
    }
    else
       Jump to Next Program Address;
}
else // Angle Comp. bit == 1 AND NAF_global == 0
   Jump to Next Program Address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.14 MOV32 (Data Move 32)

Syntax

```
MOV32 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[control={OFF | ON}]
[z_cond={OFF | ON}]
[init={OFF | ON}]| ON}]
type={IMTOREG | IMTOREG&REM | REGTOREM | REMTOREG}
[reg={A | B | R | S | T | NONE}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 17-111. MOV32 Program Field (P31:P0)

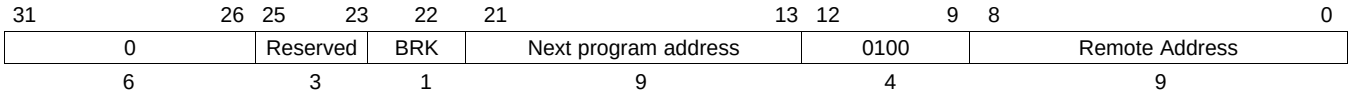


Figure 17-112. MOV32 Control Field (C31:C0)

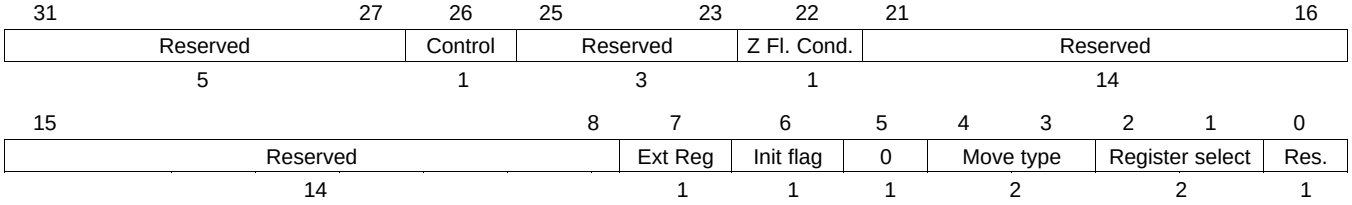
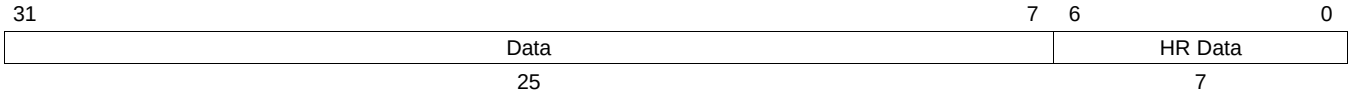


Figure 17-113. MOV32 Data Field (D31:D0)



Cycles One or two cycles

Register modified Selected register (A, B, R, S, or T)

Description MOV32 replaces the selected ALU register and/or the data field values at the remote address location depending on the move type.

Figure 17-114 through Figure 17-117 illustrate these operations. If *no register* is selected, the move is not executed, except for configuration C4:C3 = 01, where the remote data field is written with the immediate data field value.

remote Determines the location of the remote address.
Default: Current instruction + 1.

z_cond When set to OFF the MOV32 performs the move operation specified by the move type whenever it is executed (independent on the state of the Z-Flag).
When set to ON the MOV32 performs the move operation specified by the move type only when the Z-Flag is set.

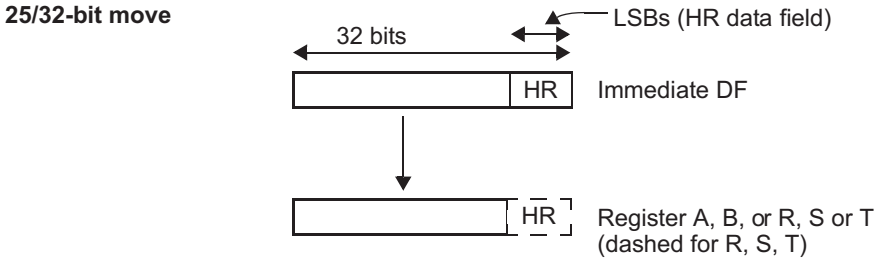
init (Optional) Determines whether or not system flags are initialized. A value of ON reinitializes the following system flags to these states:
Acceleration flag (ACF) = 0
Deceleration flag (DCF) = 1
Gap flag (GPF) = 0
New angle flag (NAF) = 0
A value of OFF results in no change to the system flags.

type Specifies the move type to be executed.

Table 17-62. Move Type Encoding Selection

Move Type	C4	C3	Source	Destination(s)	Cycles
IMTOREG	0	0	Immediate data field	Register A, B, R, S, or T	1
IMTOREG&REM	0	1	Immediate data field	Remote data field and register A, B, R, S, or T	1
REGTOREM	1	0	Register A, B, R, S, or T	Remote data field	1
REMTOREG	1	1	Remote data field	Register A, B, R, S, or T	2

Figure 17-114. MOV32 Move Operation for IMTOREG (Case 00)



reg Specifies which register (A, B, T, or NONE) is involved in the move. A register (A, B, or T) must be specified for every move type except IMTOREG&REM. If *NONE* is used with move type IMTOREG&REM, the MOV32 executes a move from the immediate data field to the remote data field. If *NONE* is used with any other move type, no move is executed.

data Specifies a 25-bit integer value to be written to the remote data field or selected register.

hr_data (Optional) HR delay. The default value for an unspecified bit is 0.

Figure 17-115. MOV32 Move Operation for IMTOREG&REM (Case 01)

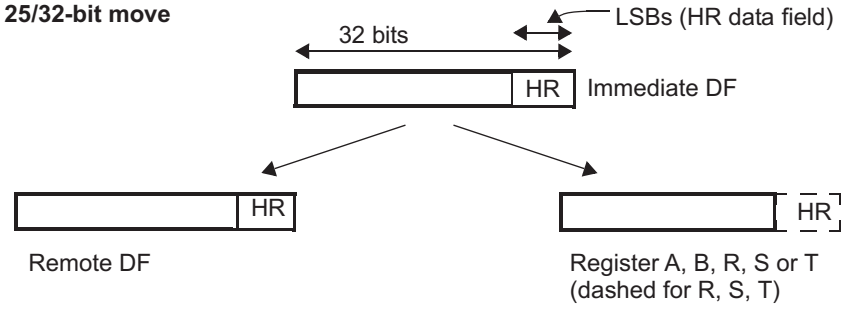


Figure 17-116. MOV32 Move Operation for REGTOREM (Case 10)

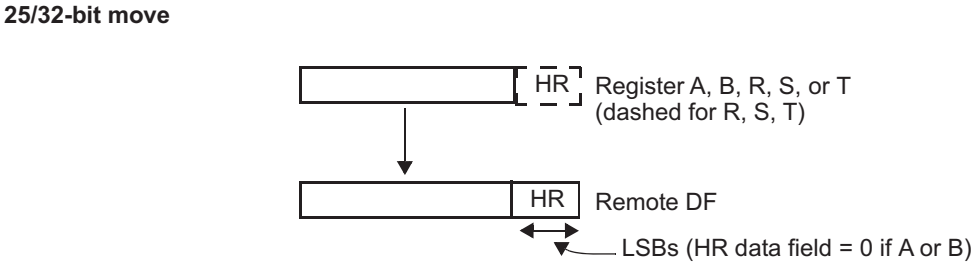
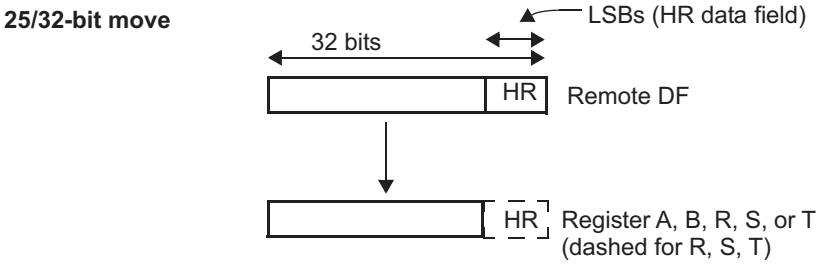


Figure 17-117. MOV32 Move Operation for REMTOREG (Case 11)



Execution

```
If [(z_cond C[22] ==0) OR ((z_cond C[22] == 1) AND (Z Flag == 1))]
{
   switch (type C[4:3])
   {
      case 00: // IMTOREG
          Selected register = Immediate Data Field;
      case 01: // IMTOREG&REM
          Selected register = Immediate Data Field;
          Remote Data Field = Immediate Data Field;
      case 10: // REGTOREM
          Remote Data Field = Selected register;
      case 11: // REMTOREG
          Selected register = Remote Data Field;
   }
}

If (Init Flag == 1)
{
   ACF = 0;
   DCF = 1;
   GPF = 0;
   NAF = 0;
}
else
  All flags remain unchanged;

Jump to Next Program Address;
```

17.5.3.15 MOV64 (Data Move 64)

Syntax

```
MOV64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
[pin={pin number}]
comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

-or-

Syntax

```
MOV64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
cntl_val={29-bit unsigned integer}
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

Figure 17-118. MOV64 Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Reserved	BRK	Next program address	0001	Remote Address
6	3	1	9	4	9

Figure 17-119. MOV64 Control Field (C31:C0)

31–29	28–27	26	25–23	22	21–16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15–13	12–8	7	6–5	4–3	2–1	0
Conditional address	Pin select	Ext Reg	Comp. mode	Action	Register select	Int. ena
9	5	1	2	2	2	1

Figure 17-120. MOV64 Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles One

Register modified None

Description This instruction modifies the data field and the control field at the remote address.

MOV64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similarly to the format of the MOV64 control field. A second syntax, in which the entire 29-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar to the MOV64 control field. Either syntax may be used, but you must use one or the either but not a combination of syntaxes. See Figure 17-121.

Figure 17-121. MOV64 Move Operation

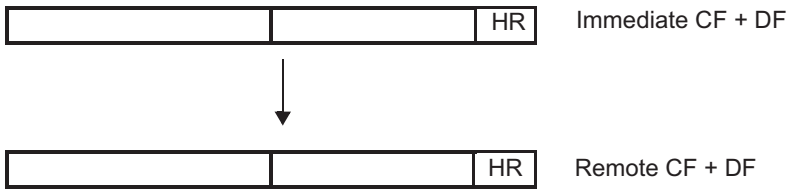


Table 17-63. MOV64 Control Field Descriptions

request	Maintains the control field for the remote instruction.
control	Maintains the control field for the remote instruction.
en_pin_action	Maintains the control field for the remote instruction.
cond_addr	Maintains the control field for the remote instruction.
pin	Maintains the control field for the remote instruction.
register, ext reg	Maintains the control field for the remote instruction.
comp_mode	Selects the comparison mode type to be used by the remote instruction.

Table 17-63. MOV64 Control Field Descriptions (continued)

action	Maintains the control field for the remote instruction.
irq	Maintains the control field for the remote instruction.
data	Specifies the 25-bit initial count value for the data field. If omitted, the field defaults to 0.
hr_data	(Optional) HR delay. The default value for an unspecified bit is 0.

Table 17-64. Comparison Type Encoding Format

comp_mode	C[6]	C[5]	MCMP Order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

Execution

```
Remote Data Field = Immediate Data Field;
Remote Control Field = Immediate control Field;
Jump to Next Program Address;
```

17.5.3.16 PCNT (Period/Pulse Count)

Syntax

```
PCNT {
[hr_lr={HIGH | LOW}]
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[irq={OFF | ON}]
type={FALL2RISE | RISE2FALL | FALL2FALL | RISE2RISE}
pin={pin number}
[control={OFF | ON}]
[prv={OFF | ON}]
[period={25-bit unsigned integer}]
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

Figure 17-122. PCNT Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8	7–6	5	4–0
0	Request Number	BRK	Next program address	0111	Int. ena	Type select	hr_lr	Pin select
6	3	1	9	4	1	2	1	5

Figure 17-123. PCNT Control Field (C31:C0)

31–29	28–27	26	25	24–0
Res.	Request type	Control	Prv.	Period Count
3	2	1	1	25

Figure 17-124. PCNT Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles One

Register modified Register A

Description This instruction detects the edges of the external signal at loop start and measures its period or pulse duration. The counter value stored in the control field C[24:0] and in the register A is incremented each N2HET loop. PCNT uses the HR structure on the pin to measure an HR period/pulse count value.

hr_lr (Optional) Specifies whether the PCNT instruction captures the HR delay into the HR data field on the selected edge condition. If hr_lr is 0 (HIGH) then PCNT captures the HR delay. if hr_lr is 1 (LOW) then PCNT only captures at loop resolution.

irq (Optional) Specifies whether or not an interrupt is generated. A value of ON sends an interrupt when a new value is captured. If OFF is selected, no interrupt is generated.

type (Optional) Determines the type of counter that is implemented.

Table 17-65. Counter Type Encoding Format

	P7	P6	Period/Pulse Select	Reset On	Capture On
FALL2RISE	0	0	Count low-pulse duration on selected pin	Falling edge	Rising edge
RISE2FALL	0	1	Count high-pulse duration on selected pin	Rising edge	Falling edge
FALL2FALL	1	0	Count period between falling edges on selected pin	Falling edge	Falling edge
RISE2RISE	1	1	Count period between rising edges on selected pin	Rising edge	Rising edge

period Specifies the 25-bit integer value that holds the counter value. The counter value is also stored in register A.
Default: 0.

data 25-bit integer representing the last captured counter value.
Default: 0.

hr_data HR delay.
Default: 0.

If *period-measure* is selected, PCNT captures the counter value into the period/pulse data field [D31:D7] on the selected edge. The HR structure provides HR capture field [D6:D0]. The counter value [C24:C0] is reset on the same edge. The captured period value is a 32-bit value.

If *pulse-measure* is selected, PCNT captures the counter value into the period/pulse count field [D31:D7] on the selected edge. The HR structure provides HR capture field [D6:D0]. The counter value [C24:C0] is reset on the next opposite edge. The captured pulse value is a 32-bit value.

When the overflow count (all 1's in the counter value) is reached, PCNT stops counting until the next reset edge is detected.

Note: For FALL2FALL/RISE2RISE, the user should always discard the first interrupt/HTU request if interrupt/request are enabled before HET_ON. For both the types, reset edge and capture edge are the same and the interrupt or HTU request is triggered on capture edge (which is nothing but the reset edge). Once the execution unit is enabled, the first edge generates an interrupt but the value of the counter is of no use as this is not the period between 2 edges. So first edge after turning on N2HET is used mainly for resetting the counter and start the period count.

Execution

```
Z = 0;

If (Period C[24:0] != 1FF_FFFFh) {
    Period C[24:0] = Period C[24:0] + 1;
}

Register A = Period C[24:0];

If (specified capture edge detected on selected pin)
{
    Z = 1;

    If (Period value != 1FF_FFFFh)
    {
       HR Capture Value = selected HR counter;
    }
    else
    {
       HR Capture Value = 7Fh;
    }

    If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
    If ([C28:C27] == 01) Generate request on request line [P25:P23];
    If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
}

If (specified reset edge detected on selected pin)
{
    Period value = 0000000h;
}

Prv bit = Current Logic (Lx) value of selected pin;

Jump to Next Program Address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.17 PWCNT (Pulse Width Count)

Syntax

```
PWCNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[hr_lr={HIGH | LOW}]
[control={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}
[en_pin_action={OFF | ON}]
pin ={pin number}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 17-125. PWCNT Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8	7–6	5–0
0	Request Number	BRK	Next program address	1010	hr_lr	11	Reserved
6	3	1	9	4	1	2	6

Figure 17-126. PWCNT Control Field (C31:C0)

31–29	28–27	26	25–23	22	21–16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15–13	12–8	7–5	4–3	2–1	0
Conditional address	Pin select	Reserved	Action	Register select	Int. ena
9	5	3	2	2	1

Figure 17-127. PWCNT Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles	One
Register modified	Selected register (A, B or T)
Description	This instruction defines a virtual timer used to generate variable length pulses. The counter value stored in the data field is decremented unconditionally on each timer resolution until it reaches zero, and it then stays at zero until it is reloaded with a non-zero value. The specified pin action is performed as long as the count after count value is decremented is greater than 0. The opposite pin action is performed when the count after decrement just reaches 0. If the hr_lr bit is reset, the opposite pin action will be taken after a HR delay from the next loop resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in bits [D6:D0].

irq	ON generates an interrupt when the data field value reaches 0. No interrupt is generated for OFF. Default: OFF.
data	25-bit integer value serving as a counter.
hr_data	HR delay. Default: 0.

Execution

```
If (Data field value == 0)
{
   Selected register = 0;
   Jump to Next Program Address;
}

If (Data field value > 1)
{
   Selected register = Data field value - 1;
   Data field value = Counter value - 1;

   If (Enable Pin action == 1)
   {
      Selected Pin = Pin Action AT next loop resolution clock;
   }

   Jump to Next Program Address;
}

If (Data field value == 1)
{
   Selected register = 0000000h;
   Data field value = 0000000h;

   If (Opposite action == 1)
   {
      If (hr_lr bit == 0)
      {
         If (Enable Pin action == 1)
         {
              Selected Pin = Opposite level of Pin Action AT next loop resolution clock
              + HR delay;
         }
      }
      else
      {
         If (Enable Pin action == 1)
         {
            Selected Pin = Opposite level of Pin Action AT next loop
            resolution clock;
         }
      }

      If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
      If ([C28:C27] == 01) Generate request on request line [P25:P23];
      If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

   }

   Jump to Conditional Address
}
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.18 RADM64 (Register Add Move 64)

Syntax

```
RADM64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
[pin={pin number}]
comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

-or-

Syntax

```
RADM64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
cntl_val={29-bit unsigned integer}
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

Figure 17-128. RADM64 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	0011	Remote Address
6	3	1	9	4	9

Figure 17-129. RADM64 Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15 13	12 8	7	6	5 4	3 2	1 0
Conditional address	Pin select	Ext Reg	Comp. mode	Action	Register select	Int. ena
9	5	1	2	2	2	1

Figure 17-130. RADM64 Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

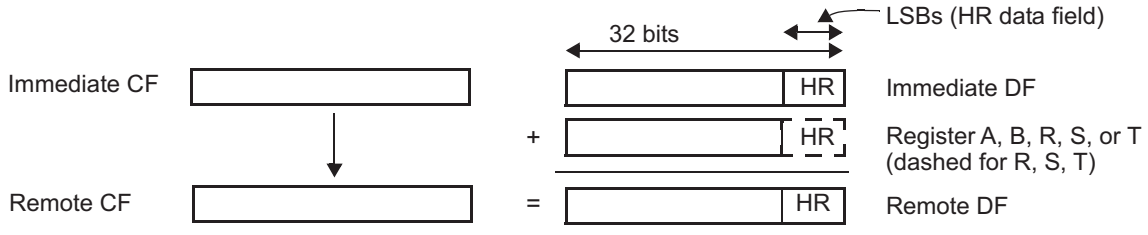
Cycles Normally One Cycle. Two cycles if writing to remote address that is also the next address.

Register modified None

Description This instruction modifies the data field, the HR data field and the control field at the remote address. The advantage over DADM64 is that It executes one cycle faster. In case the R, S, or T register is selected, the addition is a 32-bit addition. The table description shows the bit encoding for determining which ALU register is selected.

RADM64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similar to the format of the RADM64 control field. A second syntax, in which the entire 29-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar from the RADM64 control field. Either syntax may be used, but you must use one or the either but not a combination of syntaxes. See Figure 17-131.

Figure 17-131. RADM64 Add and Move Operation



comp_mode Selects the comparison mode type to be used.

Table 17-66. Comparison Type Encoding Format

comp_mode	C[6]	C[5]	MCMP Order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

Table 17-67. RADM64 Control Field Descriptions

request	Maintains the control field for the remote instruction.
Control	Maintains the control field for the remote instruction.
en_pin_action	Maintains the control field for the remote instruction.
cond_addr	Maintains the control field for the remote instruction.
pin	Maintains the control field for the remote instruction.
register	Maintains the control field for the remote instruction.
action	Maintains the control field for the remote instruction.
irq	Maintains the control field for the remote instruction.
data	Specifies the 25-bit initial value for the data field. If omitted, the field defaults to 0.
hr_data	Seven least significant bits of the 32-bit data field. Default: 0.
cntl_val	Specifies the 29 least significant bits of the Control field.

Execution

```
Remote Data Field = Selected register + Immediate Data Field (including HR field);
Remote Control Field = Immediate Control Field;
Jump to Next Program Address;
```

17.5.3.19 RCNT (Ratio Count)

Syntax

RCNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[control={OFF | ON}]
divisor={25-bit unsigned integer}
[data={25-bit unsigned integer]
}

Figure 17-132. RCNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 7	6	5 4	3 1	0
0	Reserved	BRK	Next program address	1010	Res.	00	Step width	Res.	1
6	3	1	9	4	1	2	2	3	1

Figure 17-133. RCNT Control Field (C31:C0)

31 27	26	25	24 0
Reserved	Control	Res.	Divisor
5	1	1	25

Figure 17-134. RCNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles Two Cycles (One Cycle if T=0)

Register modified None

Description RCNT is used with other instructions to convert an input period measurement T_{Input} to the form of (Equation 33) where the input period is expressed as a fraction of a reference period $T_{Reference}$.

$$T_{Input} = T_{Reference} \bullet \left(\frac{N}{M} \right) \quad (33)$$

RCNT computes the numerator N of (Equation 33). The denominator M of (Equation 33) is a constant that is of interest. For example, choosing M = 100 allows the input period to be expressed as a percentage (%) of the reference period. Note that if $T_{Input} > T_{Reference}$, then RCNT will return N > M ; which would be correct if for example the input pulse period is 110% of the reference pulse period.

RCNT expects that register T is loaded with the value of $T_{Reference}$. The input period T_{Input} is determined by counting the number of loop resolution periods between edges on the input pin. This information is conveyed through the Z flag from a PCNT instruction that precedes the RCNT instruction.

The divisor field of the RCNT instruction should be chosen as: `Divisor = M · lr` , where M is the desired denominator from (Equation 33) and lr is the loop resolution prescale value.

An example N2HET program that makes use of the RCNT instruction is:

```
L0: MOV32 { remote=dummy,type=IMTOREG,reg=T,data=0x8,hr_data=0};

L1: PCNT { hr_lr=HIGH,brk=OFF,type=FALL2FALL,pin=0};

L2: RCNT { divisor=320,data=0x4};

L3: BR {cond_addr=L5, event = Z}

L4: ADC { src1=ZERO,src2=IMM,dest=IMM,next=L0,data=0,hr_data=0};

L5: ADD { src1=REM,src2=ZERO,dest=IMM,remote=L4,data=0,hr_data=0};

L6: ADD { src1=ZERO,src2=ZERO,dest=NONE,rdest=REM,
           next=L0,remote=L4,data=0,hr_data=0};

dummy
```

In this small program an input signal on pin 0 is measured both in terms of absolute cycles by the PCNT instruction at L1 and as in 1/10ths of the reference period by the RCNT instruction at L2. In this example the reference period is a constant 0x400 cycles; this value is loaded into register T by the MOV32 instruction at L0. (0x400 is data=8, hr_data=0)

RCNT follows PCNT and is initialized to a working count of T/2 (0x200) whenever the PCNT instruction detects a falling edge on pin 0. Between falling edges on pin0, RCNT accumulates counts 10x faster than PCNT; so that the working data field of RCNT will reach the reference value of 0x400 in 1/10th the time that a PCNT instruction would. Each time the RCNT instruction passes the reference value, it sets the carry out flag and subtracts the reference value from the working count. By accumulating carry-outs from RCNT, the add with carry instruction at L4 effectively counts in increments of 1/10th of the reference period. Note that the divisor value 320 is 10 times 32; this assumes lr=32.

When the next falling edge is detected on pin 0, PCNT sets the Z flag and the RCNT instruction resets again to the initial data field of T/2. RCNT does not modify the Z flag, so that the branch instruction at L3 can execute instructions at L5, L6 instead of L4. The instructions at L5 and L6 capture the final result from L4 and reset the ADC instruction at L4 to zero for the start of the next period measurement.

Execution

```
If (register T[31:0] != 00000000h)
{
   C = 0;

   If (Z == 0)
   {
       Data Field[31:0] = Data Field[31:0] + Divisor[24:0];

       If (Data Field[31:0] >= Reg T[31:0])
       {
          Data Field[31:0]=Data Field[31:0] - Reg T[31:0];
          C = 1;
       }
   }
   else
   {
      Data Field[31:0] = T[31:0] >> 1; /* T/2 */
   }
}
Jump to Next Program Address;
```

17.5.3.20 SCMP (Sequence Compare)

Syntax

```
SCMP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
cond_addr={label | 9-bit unsigned integer}
pin ={pin number}
[action={CLEAR | SET}]
[restart={OFF | ON}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
}
```

Figure 17-135. SCMP Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Request Number	BRK	Next program address	0000	Reserved
6	3	1	9	4	9

Figure 17-136. SCMP Control Field (C31:C0)

31–29	28–27	26	25	24–23	22	21–16
Reserved	Request type	Control	Cout prv	Reserved	En. pin action	Conditional address
3	2	1	1	2	1	9

15–13	12–8	7	6–5	4	3–2	1	0
Conditional address	Pin select	Res.	01	Action	Reserved	Restart enable	Int. ena
9	5	1	2	1	2	1	1

Figure 17-137. SCMP Data Field (D31:D0)

31–7	6–0
Data	Reserved
25	7

Cycles One

Register modified Register T (implicitly)

Description This instruction alternately performs angle- and time-based operations to generate pulse sequences, using the angle referenced time base. These pulse sequences last for a relative duration using a free running time base. Generally, register B holds the angle values and register A holds the time values. Bit 0 of the conditional address field (C13) specifies whether the instruction is operating in angle or time operation mode.

When the compared values match in angle mode, a pin can be set or reset according to the pin action bit (C4). The pin does not change states if the enable pin action bit (C22) is reset.

The restart enable bit (C1) provides the option to unconditionally restart a sequence using the X-flag bit of ACMP.

restart	If restart is set to ON and the X flag = 1, the assembler writes a value of 1 into the immediate index field, writes the value in register A into the immediate data field, and jumps to the next program address. The X flag is set or cleared by the ACMP instruction. If restart is set to OFF, the X flag is ignored; no special action is performed. Default: OFF.
irq	ON generates an interrupt if the compare match occurs in angle mode. No interrupt is generated when the field is OFF. Default: OFF.
data	Specifies the 25-bit compare value.
cond_addr	Since the LSB of the conditional address is used to select between time mode and angle mode, and since the conditional address is taken only in time mode, the destination for the conditional address must be odd.

Execution

```
If (Data field value <= Selected register value) Cout = 0; else Cout = 1;

If (Restart Enable == 1 AND X == 1)
{
   C13 = 1;
   Immediate Data Field = Register A;
   Cout = 0;
   Jump to Next Program Address;
}

If (Angle Mode (C13 == 0) AND ((Restart En. == 1 AND X == 0) OR Restart En. == 0))
{
   If (Z == 0 AND (Register B value - Angle Inc. < Data field value) AND Cout == 0) OR
      (Z == 1 AND (Cout_prv == 1 OR Cout == 0)))
   {
      If (Enable Pin Action == 1) Selected Pin = Pin Action;
      If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
      If ([C28:C27] == 01) Generate request on request line [P25:P23];
      If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

      Immediate Data Field = Register A;
      C13 = 1; /*** switch to Time Mode ***/

   }
   Jump to Next Program Address;
}
Else If (Time Mode (C13 == 1)) AND ((Restart En. == 1 AND X == 0) OR Restart En. == 0)
{
   /* Result of subtract must not exceed 2^24 - 1 */
   Register T = Register A - Immediate Data Field;
   Jump to Conditional Program Address;
}
Cout_prv = Cout; (always executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.21 SCNT (Step Count)

Syntax

```
SCNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
step={8 | 16 | 32 | 64}
[control={OFF | ON}]
gapstart={25-bit unsigned integer}
[data={25-bit unsigned integer]
}
```

Figure 17-138. SCNT Program Field (P31:P0)

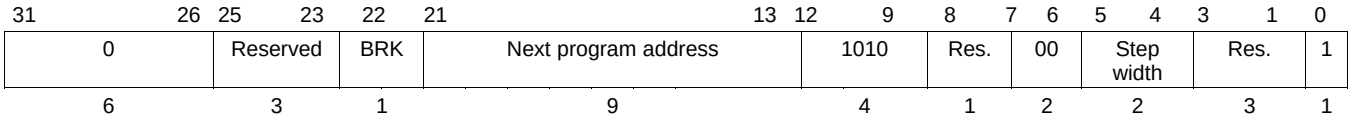
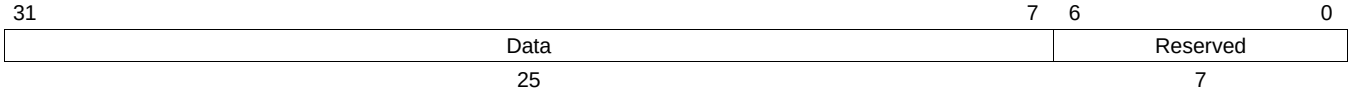


Figure 17-139. SCNT Control Field (C31:C0)

Figure 17-140. SCNT Data Field (D31:D0)



Cycles One or two cycles (two cycles when DF is involved in the calculations)

Register modified Register A

Description This instruction can be used only once in a program and defines a specialized virtual timer used after APCNT and before ACNT to generate an angle-referenced time base synchronized to an external signal (that is, a toothed wheel signal) as defined in APCNT and ACNT. Step width selection bits are saved in two flags, SWF0, and SWF1, to be re-used in ACNT.

SCNT multiplies the frequency of the external signal by a constant *K* defined in the step width field, [P5:P4]. The bit encoding for this field is defined in Table 17-68.

step Specifies the step increment to be added to the counter value each program resolution. These two bits provide the values for the SWF0 and SWF1 flags. The valid values are listed in Table 17-68.

Table 17-68. Step Width Encoding for SCNT

P5	P4	Step Width (K)
0	0	8
0	1	16
1	0	32
1	1	64

gapstart Defines the gap start angle, which SCNT writes to register A. The gap start value has no effect on the SCNT instruction, but if the ACNT instruction is being used, register A must contain the correct gap start value. For a typical toothed wheel gear:

GAPSTART = (*stepwidth* × (*actual teeth on gear* - 1)) + 1.

data Specifies the 25-bit integer value serving as a counter.

Default: 0.

This instruction is incremented by the step value K on each timer resolution up to the previous period value P(n-1) measured by APCNT (stored in register T). The resulting period of SCNT is: P(n - 1)/K

Due to stepping, the final count of SCNT will not usually exactly match the target p(n-1). SCNT compensates for this error by starting each cycle with the remainder of the previous cycle.

When SCNT reaches the target p(n-1), the zero flag is set as an increment condition for ACNT.SCNT also specifies a gap start angle, defining the start of a range in ACNT where period measurements in APCNT are temporarily stopped to mask singularities in the external signal.

SCNT uses register A to store the gap start value. Gap start has no effect for SCNT.

Execution

```
SWF1 = P5;
SWF0 = P4;
Z = 0;

If (register T != 0000000h)
{
   If (DCF == 1 OR ACF == 1)
   {
      Data Field register = 0000000h;
      Counter value = 0000000h;
   }

   If (DCF == 0 AND ACF == 0)
   {
      Data Field register = Data field register + Step Width;
   }

   If ((Data Field register - register T) >= 0)
   {
      Data field register = Data Field register - register T;
      Z = 1;
   }

   Register A = Gap start value;

}

Jump to Next Program Address;
```

17.5.3.22 SHFT (Shift)

Syntax

```
SHFT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
smode={OR0 | OL0 | OR1 | OL1 | ORZ | OLZ | IRM | ILL | IRZ | ILZ}
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}
cond={UNC | FALL | RISE}
pin ={pin number}
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
}
```

Figure 17-141. SHFT Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–4	3–0
0	Request Number	BRK	Next program address	1111	Reserved	Smode
6	3	1	9	4	5	4

Figure 17-142. SHFT Control Field (C31:C0)

31–29	28–27	26	25	24–22	21–16
Reserved	Request type	Control	Prv.	Reserved	Conditional address
3	2	1	1	3	9

15–13	12–8	7	6–5	4	3	2–1	0
Conditional address	Pin select	Ext Reg	Shift condition	Res.	0	Register select	Int. ena
9	5	1	2	1	1	2	1

Figure 17-143. SHFT Data Field (D31:D0)

31–7	6–0
Data	Reserved
25	7

Cycles One

Register modified Selected register (A, B, R, S or T)

Description This instruction shifts the data field of the Instruction. N2HET pins can be used for data in or data out. SHFT includes parameters to select the shift direction (in, out, left, right), shift condition (shift on a defined clock edge on HET[0] or shift always), register for data storage (A, B, R, S or T), and the data pin.

smode Shift mode

Table 17-69. SHIFT MODE Encoding Format

smode	P3	P2	P1	P0	Operation	
OR0	0	0	0	0	Shift Out / Right	LSB 1st on HETx / 0 into MSB
OL0	0	0	0	1	Shift Out / Left	MSB 1st on HETx / 0 into LSB
OR1	0	0	1	0	Shift Out / Right	LSB 1st on HETx / 1 into MSB
OL1	0	0	1	1	Shift Out / Left	MSB 1st on HETx / 1 into LSB
ORZ	0	1	0	0	Shift Out / Right	LSB 1st on HETx / Z into MSB
OLZ	0	1	0	1	Shift Out / Left	MSB 1st on HETx / Z into LSB
IRM	1	0	0	0	Shift In / Right	HETx into MSB
ILL	1	0	0	1	Shift In / Left	HETx into LSB
IRZ	1	0	1	0	Shift In / Right	HETx in MSB / LSB into Z
ILZ	1	0	1	1	Shift In / Left	HETx in LSB / MSB into Z

cond Specifies the shift condition.

Table 17-70. SHIFT Condition Encoding

C6	C5	Shift Condition
0	X	Always
1	0	Rising edge of HET[0]
1	1	Falling edge of HET[0]

irq ON generates an interrupt if the Z flag is set. A value of OFF does not generate an interrupt.
Default: OFF.

data Specifies the 25-bit value for the data field.

Execution

```
If (SHIFT condition == 0X)
OR (SHIFT condition == 10 AND HET[0] rising edge)
OR (SHIFT condition == 11 AND HET[0] falling edge)
{
   If ([P3:P2] == 00)
   {
       If ((Immediate Data Field == all 0’s AND [P3:P0] == 000X)
           OR (Immediate Data Field == all 1’s AND [P3:P0] == 001X))
       {
          Z = 1;
       }
       else
       {
          Z = 0;
       }
   }
   else If ([P3:P0] == 1010)
   {
      Z = LSB of the Immediate Data Field;
   }
   else if ([P3:P0] == 1011)
   {
      Z = MSB of the Immediate Data Field;
   }
}

If( (Immediate Data Field == all 0’s) OR
    (Immediate Data Field == all 1’s))
{
   if (Interrupt Enable == 1)
                                          {
      HETFLG[n] = 1;      /* n depends on address */
   }
   Jump to Conditional Address;
}
else
{
   Jump to Next Program Address;
}

Prv. bit = HET[0] Pin level; (Always executed)

Shift Immediate Data Field once according to P[3:0];

Immediate Data Field = Result of the shift;

Selected register = Result of the shift;

Jump to Next Program Address;
```

NOTE: The immediate data field evaluates all 0s or all 1s and is performed before the shift operation.

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.23 WCAP (Software Capture Word)

Syntax

```
WCAP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[hr_lr={HIGH | LOW}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin ={pin number}
event={NOCOND | FALL | RISE | BOTH}
reg={A | B | R | S | T | NONE}
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 17-144. WCAP Program Field (P31:P0)

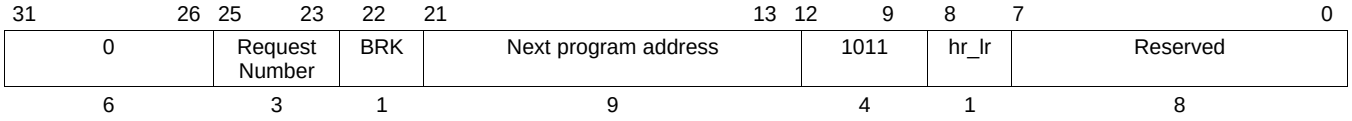


Figure 17-145. WCAP Control Field (C31:C0)

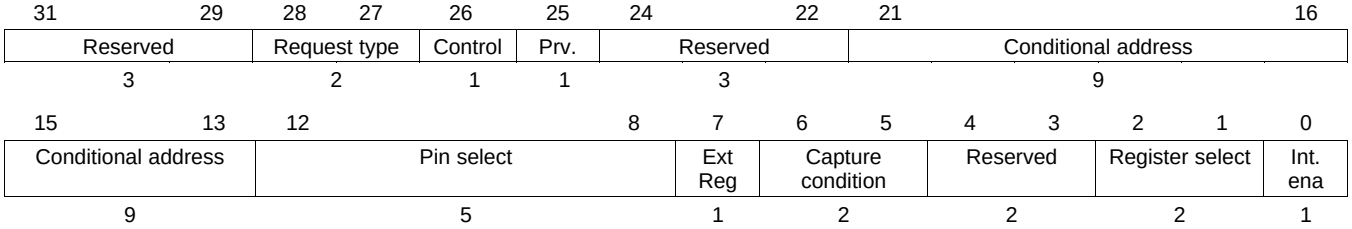
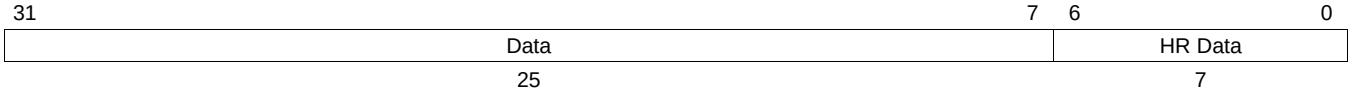


Figure 17-146. WCAP Data Field (D31:D0)



Cycles One

Register modified None

Description This instruction captures the selected register into the data field if the specified capture condition is true on the selected pin. This instruction can be used with all pins.

If the hr_lr bit is reset, the WCAP instruction will capture an HR time stamp into the data field on the selected edge condition. If the hr_lr bit is set, the HR capture is ignored.

event Specifies the event that triggers the capture.

Table 17-71. Event Encoding Format for WCAP

C6	C5	Capture Condition
0	0	Always
0	1	Capture on falling edge
1	0	Capture on rising edge
1	1	Capture on rising and falling edge

irq ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF.
Default: OFF.

data Specifies the 25-bit integer value to be written to the data field or selected register.

hr_data HR capture value.
Default: 0.

NOTE: WCAP in HR Mode: The HR Counter starts on a WCAP instruction execution (in the first loop clock) and will synchronize to the next loop clock. When N2HET is turned on and a capture edge occurs in the first loop clock (where the HR counter hasn't been synchronized to the loop clock), then the captured HR counter value is wrong and is of no use. So the captured HR data in the first loop clock should be ignored.

Execution

```
If (Specified Capture Condition is true on Selected Pin
OR Unconditional capture is selected)
{
   Immediate Data Field = Selected register value;

   If (hr_lr bit == 0) Capture the HR value in Immediate HR Data Field;

   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on request line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

   Jump to Conditional Address;
}

Jump to Next Program Address;

Prv bit = Current Logic (Lx) value of selected pin; (always executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

17.5.3.24 WCAPE (Software Capture Word and Event Count)

Syntax

```
WCAPE {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}
pin ={pin number}
event={NOCOND | FALL | RISE | BOTH}
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[ts_data={25-bit unsigned integer]
[ec_data={7-bit unsigned integer}]
}
```

Figure 17-147. WCAPE Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Request Number	BRK	Next program address	1000	Reserved
6	3	1	9	4	9

Figure 17-148. WCAPE Control Field (C31:C0)

31–29	28–27	26	25	24–23	22–16
Reserved	Request type	Control	Prv.	Reserved	Conditional address
3	2	1	1	3	9

15–13	12–8	7	6–5	4–3	2–1	0
Conditional address	Pin select	Ext Reg	Capture condition	Reserved	Register select	Int. ena
9	5	1	2	2	2	1

Figure 17-149. WCAPE Data Field (D31:D0)

31–7	6–0
Time Stamp	Edge Counter
25	7

Cycles One

Register modified None

Description This instruction captures the selected register into the data field [D31:D7] and increments an event counter [D6:D0] if the specified capture condition is true on the selected pin. This instruction can be used with all pins, but the time stamp [D31:D7] has loop resolution only.

event Specifies the event that triggers the capture.

Table 17-72. Event Encoding Format for WCAPE

C6	C5	Capture Condition
0	0	Always
0	1	Capture on falling edge
1	0	Capture on rising edge
1	1	Capture on rising and falling edge

irq ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF.
Default: OFF.

ts_data Specifies the 25-bit integer value for [D31:D7]
Default: 0.

ec_data Specifies the initial 7-bit integer value for [D6:D0].
Default: 0.

Execution

```
If (Specified Capture Condition is true on Selected Pin
OR Unconditional capture is selected)
{
   Immediate Data Field[31:7] = Selected register value;
   Immediate Data Field [6:0] = Immediate Data Field [6:0] + 1;

   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on request line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

   Jump to Conditional Address;
}

Jump to Next Program Address;

Prv bit = Current Logic (Lx) value of selected pin; (always executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 17.2.7.

High-End Timer Transfer Unit (HTU) Module

This chapter describes the high-end timer transfer unit (HTU) module. The HTU is specialized to transfer N2HET (High End Timer) data to or from the microcontroller RAM.

Topic **Page**

18.1 Overview

The HET transfer unit is a dedicated direct memory access controller that transfers data between the N2HET RAM and RAM buffers located in the main memory address range. This eliminates time consuming CPU accesses to the N2HET RAM to gather measurement data or creating output waveforms and thus freeing up the CPU to perform other tasks.

18.1.1 Features

- Independently transfers data between the N2HET and the main memory
- 8 double control packets supporting dual buffer configuration
- Transfer requests generated by N2HET instructions/events
- One shot, circular and auto switch buffer transfer modes for each double control packet for flexible buffer handling
- Constant and post-increment addressing modes
- 32- or 64-bit transactions
- Programmable memory protection region
- Parity protect control packet RAM
- Extensive diagnostic functionality

18.2 Module Operation

The HTU is tightly coupled to the N2HET and is not intended to transfer data from other peripheral modules. It initiates transfers with the help of requests generated by the N2HET program and configurable control packets. Figure 18-1 shows a system block diagram of the HTU and the main path for the data transfer. The tight coupling and the dedicated bus into the SCR (Switched Central Resource) reduces the amount of data transferred on the peripheral bus, which increases the overall system performance. However if the application decides to use the direct CPU access method to the N2HET RAM, it is free to do so.

Figure 18-2 shows a more detailed block diagram of the HTU module.

Transfers between N2HET RAM and the main memory are triggered by 8 different normal N2HET requests. Quiet requests are used for specific cases and are discussed in Section 18.2.4.1. Control packets, which store the source and destination addresses, the transfer count and other information (see Section 18.5), are associated with the requests. A FIFO decouples the read- and write-path and allows to do data-packing in the case of different read- and write-data sizes. The application can specify a section of memory into or from which the data is transferred. This serves as memory protection in the case that information in the control packet RAM was unintentionally altered and avoids that the HTU can overwrite important application data.

Control packets are implemented as double control packets (DCP) that allow to specify two buffers for the data transfer. This enables the CPU to work with one buffer, while new data is transferred to/from the other buffer.

The control packet defines:

- the start address of the source/destination buffers
- the N2HET instruction address location
- how many elements need to be transferred per request
- the buffer size as the number of elements times the number of frames
- the buffer handling

Figure 18-1. System Block Diagram

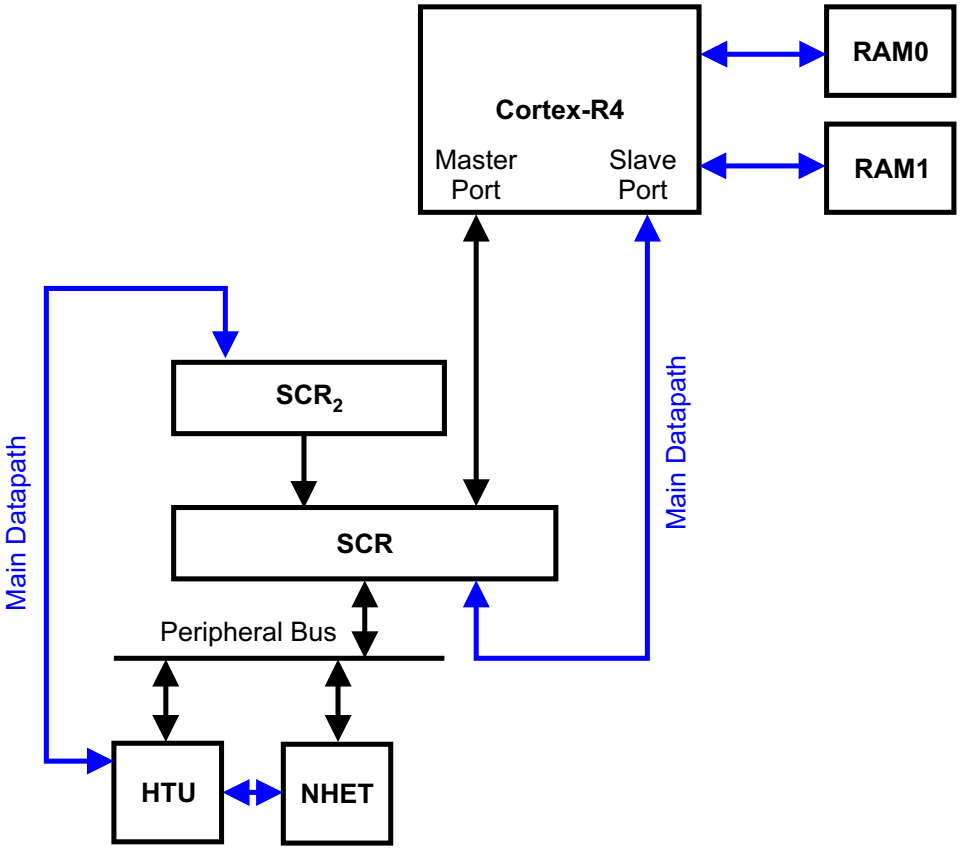
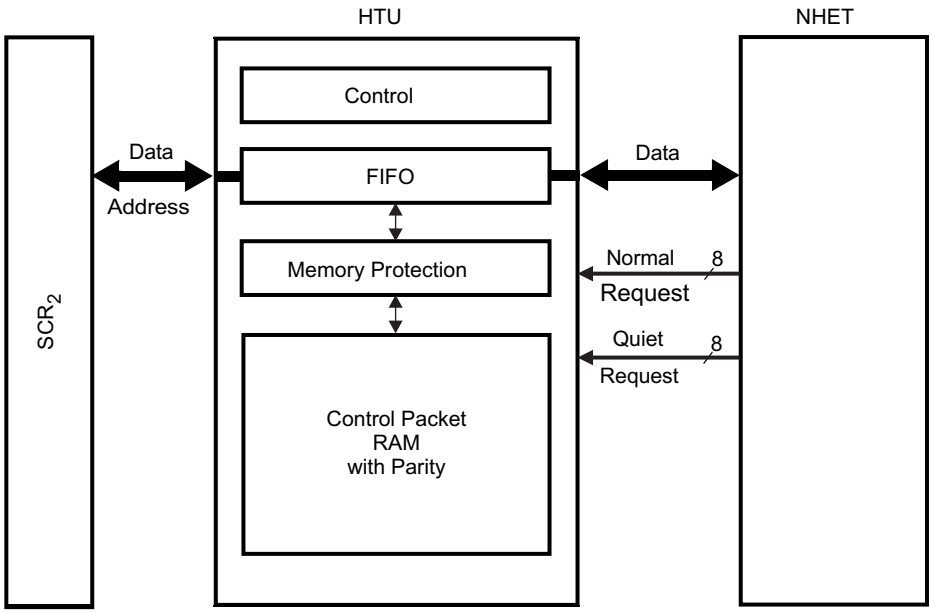
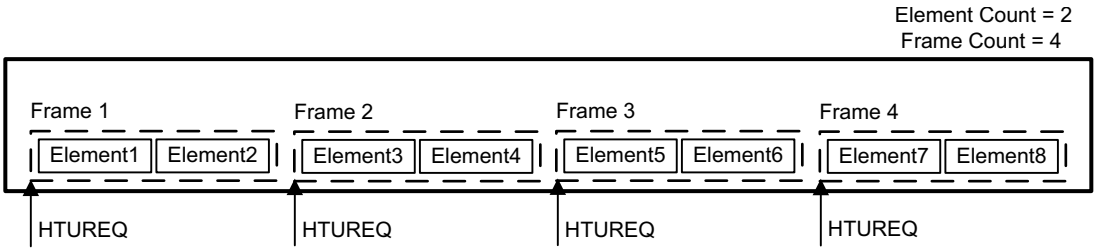


Figure 18-2. HTU Block Diagram



A transfer is triggered when a certain condition (for example, capture, compare condition) is detected by a N2HET instruction. The N2HET instruction specifies which request line to the HTU will be triggered at the event. The DCPs have a fixed assignment to the request lines and the corresponding assignment can be found in the device datasheet. Once a request is triggered, it starts a frame transfer. A frame can contain one or more elements. Elements are defined as 32-bit or 64-bit words of data.

Figure 18-3. Example of a HTU Transfer



18.2.1 Data Transfers Between Main RAM and N2HET RAM

18.2.1.1 Addressing Modes

The addressing modes of a control packet need to be distinguished between the main RAM of the CPU and the N2HET RAM.

Main RAM

For each double control packet (see Section 18.2.1.3) the addressing mode for the main RAM (RAM0/1) can be configured to constant or post-increment mode in register IHADDRCT.

- **Constant Addressing:** In constant mode, the HTU writes/reads the data to/from the same address in the main RAM.
- **Post-increment Addressing:** In post-increment mode, the HTU writes/reads the data to/from the main RAM by incrementing through the addresses after each transfer. If 32-bit transfers are selected it will automatically increment by 4 Byte, if 64-bit transfers are selected, it will increment by 8 Byte. The examples of Use Cases illustrate the post-increment mode, where the elements of consecutive frames are transferred to/from consecutive locations in the main RAM buffer.

N2HET RAM

How a DCP addresses the N2HET RAM is determined by the initial N2HET address, the initial element counter (IETCOUNT) and the N2HET addressing mode (ADDMH). The main difference to the main RAM addressing mode is that the HET address is reset to the initial HET address for every first element of a frame. To implement constant addressing, the initial element counter needs to be set to 1. Post-increment addressing is selected by programming the initial element counter to a value other than 1.

18.2.1.2 Single Buffer Implementation

In a single buffer implementation, the DCP is set up to transfer data to/from a single buffer in the main RAM. With each transfer request, the programmed number of elements is transferred and the buffer pointer is reset to its starting address after the programmed number of frame transfers have completed.

Figure 18-4 shows the request on one request line of the HTU and the frame running on the assigned control packet visualized by the element counter. In the diagram, the frame has 5 element transfers (element count = 5).

Before the application reads the buffer, it has to disable the control packet to avoid that new data overwrites the buffer while it's being accessed by the application. Regardless of the control packet being disabled at t1 or t2 the last frame will always be completed, since the trigger request has been received already. The application can determine any ongoing transfers by the TIPF flag and the NACP bits.

- **One Shot Buffer Mode:** If TMBA or TMBB is set to one shot buffer mode then the data stream will stop after all elements of buffer A or buffer B have been transferred. This means that the corresponding DCP will be disabled after the last frame was transferred to/from buffer A or B and CFTCTA or CFTCTB decrements to 0.
- **Circular Buffer Mode:** If TMBA or TMBB is set to circular buffer mode, then the data stream will continue back at the start of buffer A or B after all elements of buffer A or B have been transferred.The example of Timing Example for Circular Buffer Mode assumes IETCOUNT=3 (Initial Element Transfer Count), IFTCOUNT=3 (Initial Frame Transfer Count, SIZE=0 (Size of Transfer = 32-bit) and ADDFM=0 (Addressing Mode Main Memory = Post Increment). So there are in total 9 32-bit values in the buffer. It also assumes IFADDRx=10h. "U" means uninitialized.

Figure 18-4. Single Buffer Timing and Memory Representation

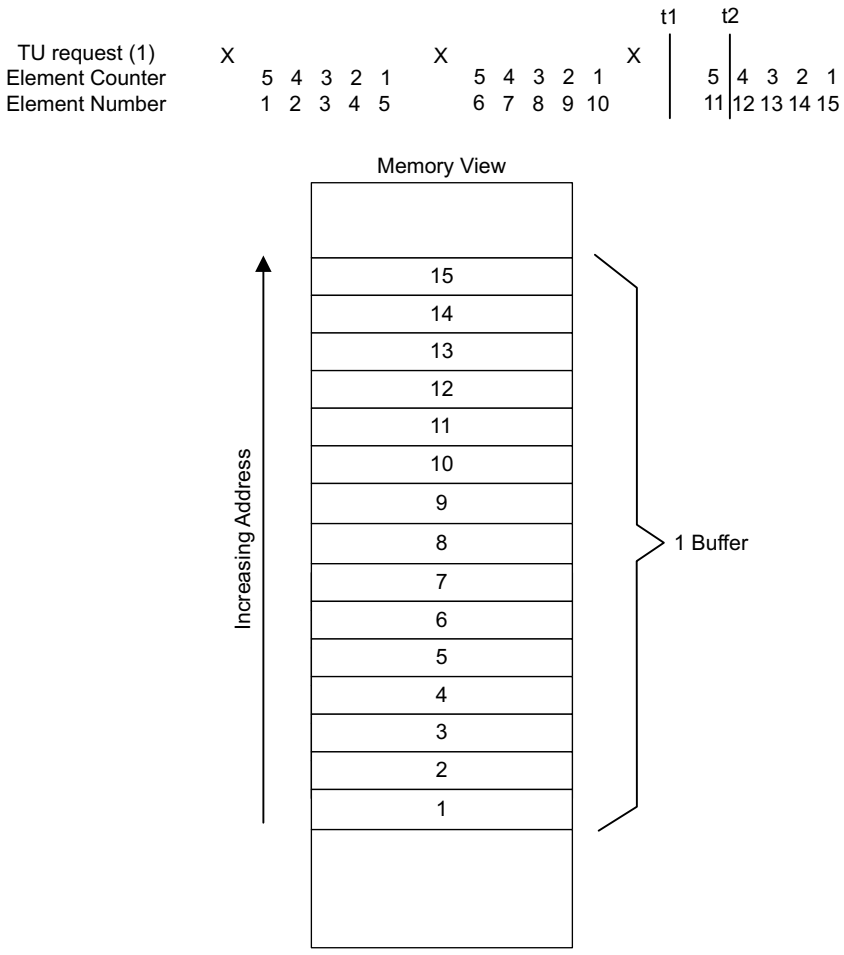
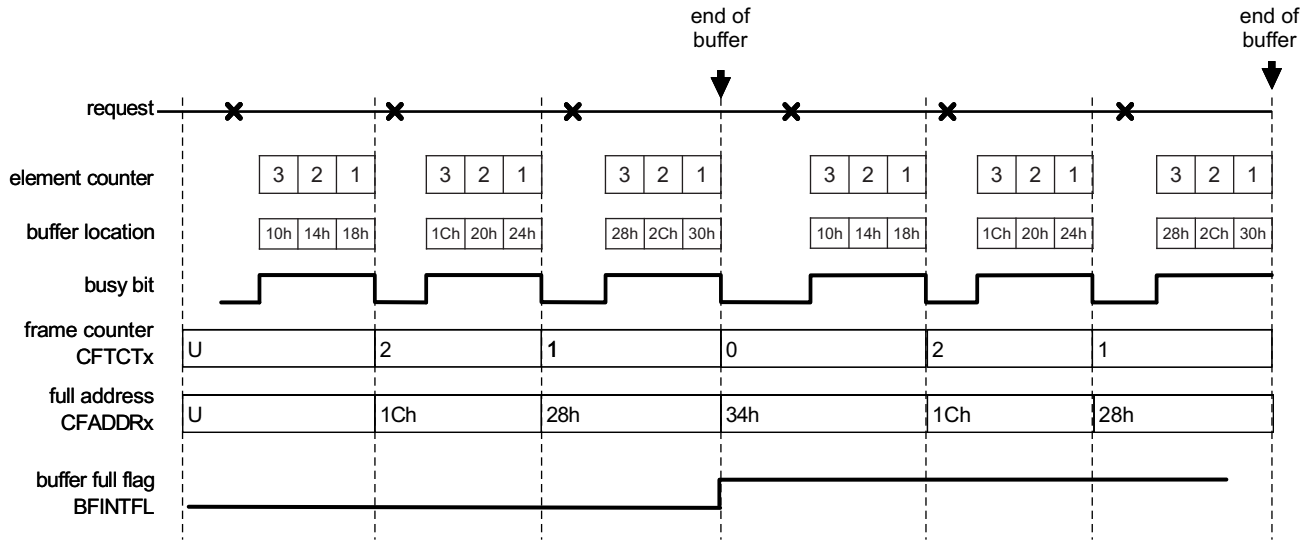


Figure 18-5. Timing Example for Circular Buffer Mode

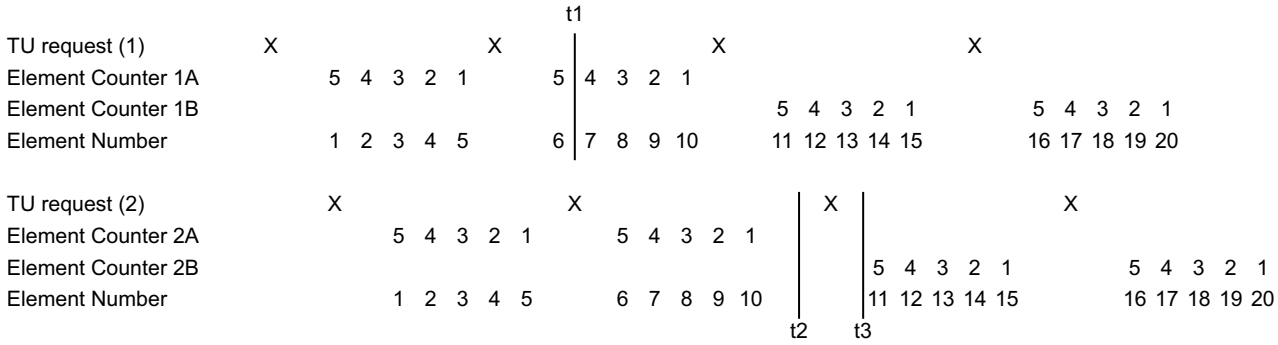


18.2.1.3 Dual Buffer Implementation

The transfer unit provides **double control packets (DCPs)** supporting the use of two buffers per data stream (per HTU request source). If one buffer should be read by the CPU, the data stream is directed to the other buffer and the first buffer is frozen. Switching to the other buffer can be triggered with a write access to the CPENA register or with the DCP configured to automatically switch to the other buffer when the programmed number of frames has been transmitted. Freezing the buffer avoids this buffer to be overwritten with new HET data while the CPU reads this buffer.

Figure 18-6 shows a timing example of two HET instructions 1 and 2, which are the request sources for the HTU (and are controlled by DCP 1 and DCP 2). Each generated frame has 5 element transfers. Request source 1 has two RAM buffers, controlled by two control packets 1A and 1B. Request source 2 has two RAM buffers, controlled by two control packets 2A and 2B.

Figure 18-6. Dual Buffer Timing



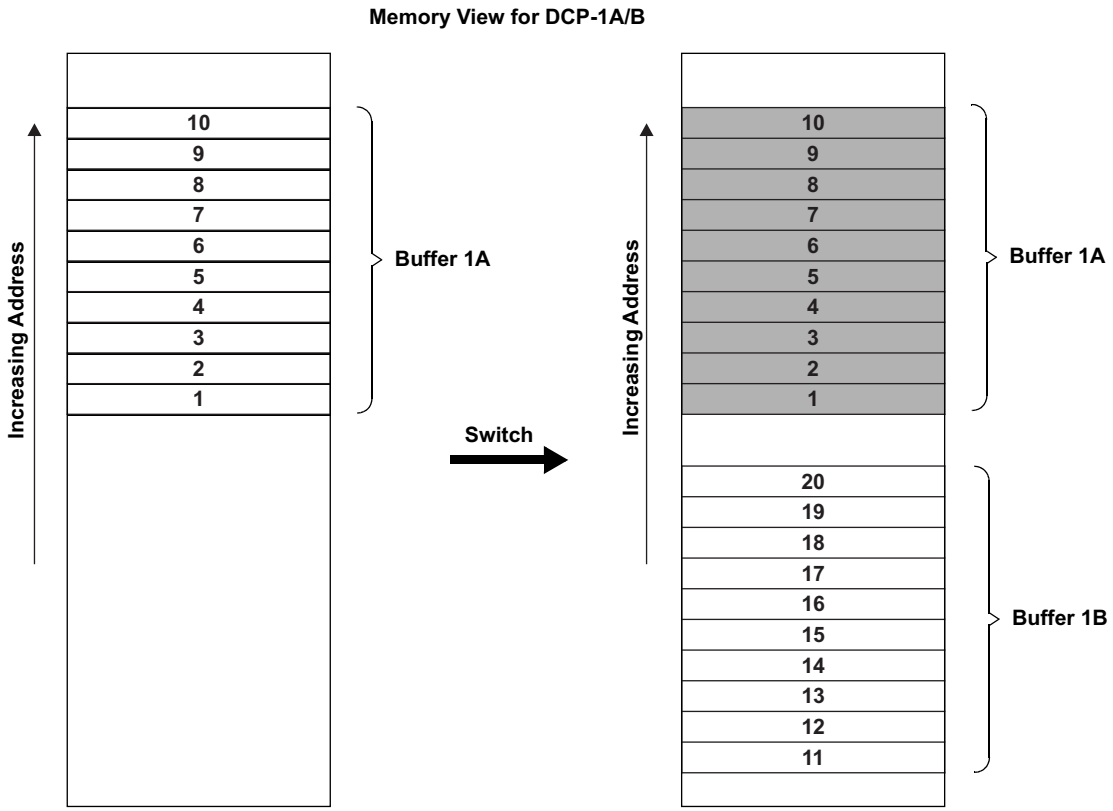


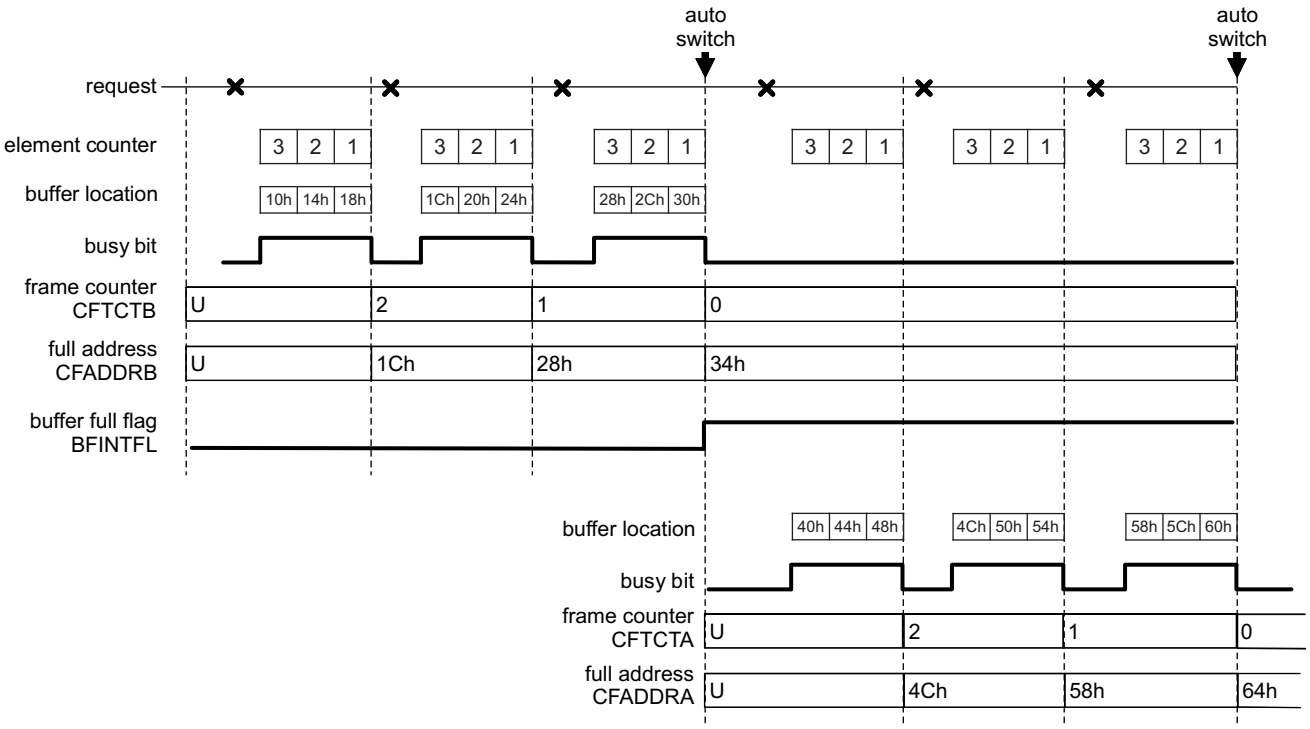
Figure 18-6 shows a switch at time t1, where buffer 1A is frozen and data stream 1 is directed to buffer 1B, but only after the frame has been completed. It also shows the time (t2 or t3) where 2A is frozen and data stream 2 is directed to buffer 2B. If the switch happens between the request and the start of the frame (for example, time t3), then the frame is processed by the new control packet (although the old control packet was active at the time of the request). The delays between the HTU requests and the start of the element transfers result from the fact that the HTU can process only one transfer at a time.

Auto Switch Buffer Mode

If TMBA is set to auto switch mode, then the data stream will continue at the start of buffer B after all elements of buffer A have been transferred. This means that in the CPENA register, CP A is disabled and CP B is enabled automatically and buffer B uses its initial main memory address and initial frame counter to start. The same principle is valid for TMBB and buffer B.

The examples of Figure 18-7 assumes IETCOUNT=3 (Initial Element Transfer Count), IFTCOUNT=3 (Initial Frame Transfer Count, SIZE=0 (Size of Transfer = 32-bit) and ADDFM=0 (Addressing Mode Main Memory = Post Increment). So there are in total 9 32-bit values in buffer A and B. It also assumes IFADDRB=10h and IFADDRA=40h. "U" means uninitialized.

Figure 18-7. Timing Example for Auto Switch Buffer Mode



18.2.1.4 General Control Packet Behavior

The action defined by the selected mode will be performed at the end of the last frame, which has the frame counter value of 1. The one shot and auto switch mode will automatically update the CPENA register at this time. Note, that for all three modes listed above, it is possible to switch to the other buffer by writing to CPENA before the end of the current buffer is reached.

If a write access to CPENA happens while the last frame of DCP x (with frame counter = 1) is transferred then the priority is defined by Table 18-1.

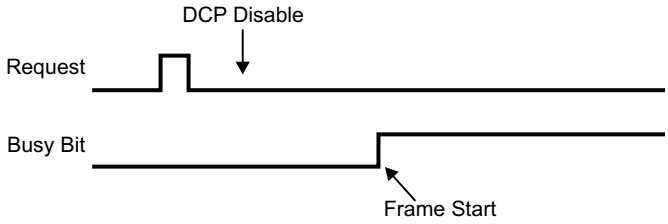
Table 18-1. CPENA / TMBx Priority Rules

Write access to CPENA bits (2*x+1) and (2*x) during the frame with frame counter = 1 ([1])	Priority Rule
Disable: 01 --> 00 or 10 --> 00	Disabling the DCP by the write to CPENA has priority, TMBx is ignored.
Stay: 01 --> 01 or 10 --> 10	The write access to CPENA is ignored, TMBx has priority and defines the action.
Switch: 01 --> 10 or 10 --> 01	Switching the DCP by the write to CPENA has priority, TMBx is ignored.

(1) See read table of CPENA register (Table 18-14)

There could be a case where the CPU wants to do main memory operations, but does not want the HTU modifying the main memory. It could happen that a request was already active, but the frame transfer hasn't started yet when the application disabled the control packets. The timing diagram in Figure 18-8 shows this scenario.

Figure 18-8. Timing for Disabling Control Packets



Since the request for the transfer was already received before the DCPx is disabled, the HTU will still start the frame transfer. The application would poll the BUSYx bit during the time the DCPx was disabled and before the frame was started and would read a non-busy information. It then would start the main memory operations thinking all transfers have completed, however after some time the HTU will start the outstanding frame transfer and corrupt the main memory.

To avoid this, the application can set the VBUSHOLD bit to disable all transactions between the HTU and the main memory. It has to poll the BUSBUSY bit to ensure that no outstanding transactions on the bus are pending. The HTU will still receive all transfer requests from the N2HET, but it will not be able to transfer any data to or from the main memory, while the VBUSHOLD bit is set.

18.2.2 Arbitration of HTU Elements and Frames

- Frames do not interrupt each other. If a request occurs on DCP x while another frame runs on DCP y (and x ≠ y), then the current frame completes before the new frame starts.
- If two or more request lines are active, the request line with the lower number (specified in the request number field of the corresponding N2HET instruction) is serviced first.

18.2.3 Conditions for Frame Transfer Interruption

If a frame is currently transferred on DCP x and one of the events listed below happens, then the event will (1.) clear the element counter of DCP x, (2.) stop new element transfers on DCP x (3.) clear the active busy bit of DCP x and (4.) disable DCP x in the CPENA register. The DCPs other than DCP x will not be affected.

- Request Lost Error of DCP x (with CORL bit set to 0).
- Parity Error of DCP x (with parity check enabled and COPE bit set to 0). See also Section 18.2.6.
- Bus Error of DCP x.
- Memory Protection Error of DCP x (with memory protection enabled). See also Section 18.2.5.
- Writing a 1 to a BUSY bit (belonging to DCP x) if that bit is 1. There is no effect if the BUSY bit is 0.
- Writing a 1 to the HTURES bit.

When a memory protection error occurs, the access to the protected address is blocked. The frame is stopped before the element, which caused the violation transfer, starts. All other errors will let the current element transfer finish.

In case of the Request Lost and Bus Error, one more element transfer goes on the bus, before the frame is actually stopped. Accordingly, the busy bit is cleared after the element, which follows the element that caused the error.

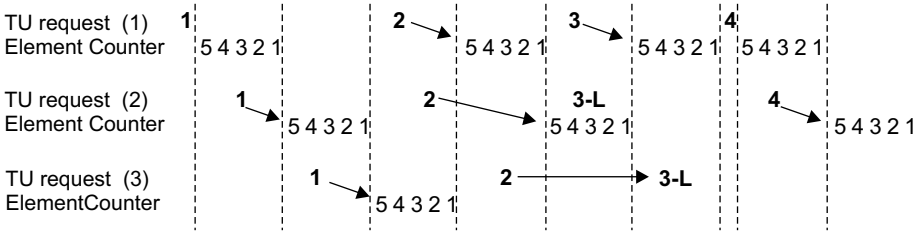
In case of the Bus Error, the counter for the element, which follows the element that caused the error, is captured to the ERRETC register field.

NOTE: If the HTUEN bit is cleared during a frame is transferred, then the frame will be completed before the HTU is disabled.

18.2.4 HTU Overload and Request Lost Detection

If the number of different HTU request sources is "high", the period between the requests is "short" and/or the initial element counter values are "big", then the HTU could get into a overload situation. In Figure 18-9, all requests marked with "L" are lost, since their frame is not completed at the time the next request occurs. Each number in the rows "TU request (x)" represents a time, where the associated N2HET instruction generates a request on DCP x. The arrows in Figure 18-9 point to the associated frame, which could be delayed compared to the request. The delays are caused by different frames, which are currently processed. The figure assumes that the CORL bit in the RLBECTRL register is set, which causes the DCP to stay enabled and let the data stream continue after a request lost error occurred on the DCP (see 3-L for TU request (2)).

Figure 18-9. Timing Example Including Lost Requests



Lost requests are signaled with the RLOSTFL register, and if enabled, can generate request lost interrupts.

If the CORL bit is set, a frame will be completed and the corresponding DCP stays enabled even if a request lost was generated during this frame.

In dual buffer mode, the request lost detection works continuously, independent of the CP switches.

18.2.4.1 Requests and Quiet Requests

In addition to generating too many transfer requests and thus overloading the HTU and not being able to transfer data at all, it can happen that inconsistent data is transferred. The following examples illustrate such scenarios.

In the examples below, the HTU reads a frame of three elements from the datafield of three different instructions. In Figure 18-10, the L3-Instruction generates the HTU request at time t2, t7, and so on. and the according frame (at t3). The frame is delayed because of the HTU load. However, as shown in Figure 18-10, the delay still allows the frame to complete before the datafield of instruction L1 is updated again. However, when the delay is longer (as shown in Figure 18-11), then the frame could fall into the N2HET loop (LRP), in which the N2HET updates the data fields of the L1, L2 and L3 instructions. In this case, the HTU could read inconsistent data as shown in the diagram. A wrong (new) value is read from L1 (at time t3), but correct ("old") values are read from L2 and L3 (at times t4 and t5).

Figure 18-10. Timing which Generates No Request Lost Error

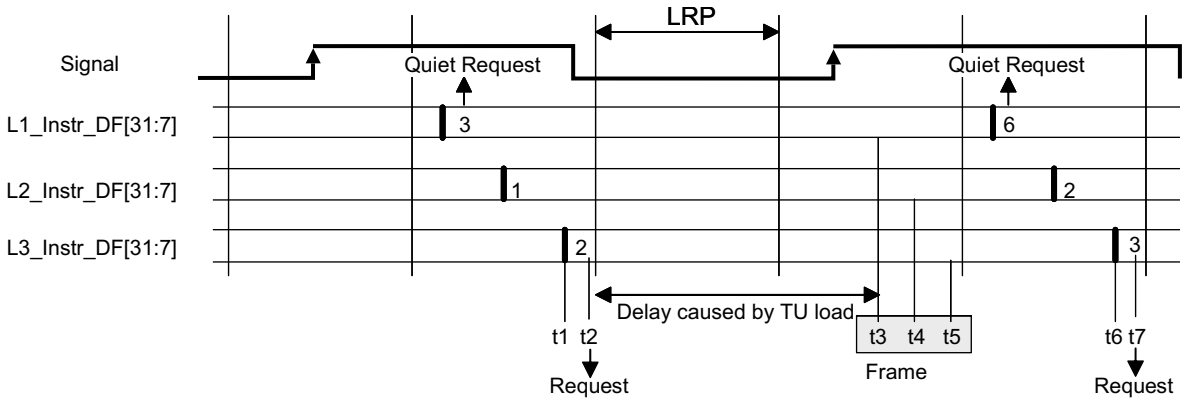
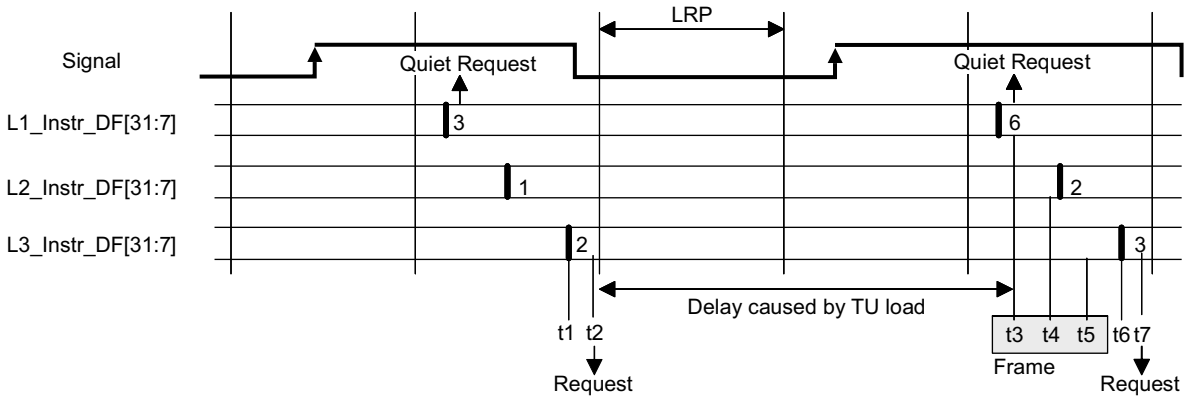


Figure 18-11. Timing which Generates a Request Lost Error



To prevent sending inconsistent data, the N2HET instructions are able to generate a **quiet request**, which does not originate a transfer but is only used by the HTU for consistency check. If a frame has not completed since the last request (or has not even started) at the time the quiet request occurs, then the HTU signals a request lost error. All instructions, which allow to generate a request can be configured to generate a quiet request instead. So in the examples of Figure 18-10 and Figure 18-11, instruction L1 should be configured to generate a quiet request and instruction L3 to generate a normal request. In the case of Figure 18-11, the corresponding bit in the RLOSTFL register will be set.

It is the responsibility of the N2HET software to enable a quiet request for the first instruction of an instruction block, which is addressed by DCP x, and to enable a normal request only for the last instruction of this block. Since enabling the quiet request should enable a proper request lost detection for DCP x, both N2HET instructions need to specify the same DCP x (reqnum=x).

The control fields of the HET instructions provide a 2-bit field to configure one of the following possibilities (as shown in Table 18-2). A 3-bit field in the program field will select which of the 8 Double Control Packets will be triggered by the request.

Table 18-2. Triggered Control Packets

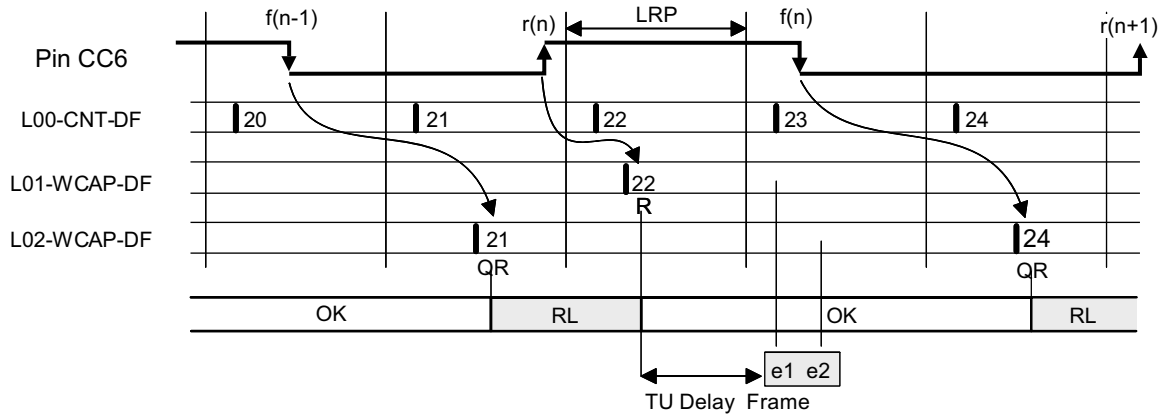
Request Type Bit 1	Request Type Bit 0		Request Number
Don't care	0	No request	Specify number 0, 1,... or 7, which selects the HTU request line
0	1	Generate normal request	
1	1	Generate quiet request	

In the case of very light HTU load, but higher signal requirements (for example, high frequency), the quiet request could also be used to define periods in which the data read by a control packet is safe. The following HET code will capture counter time stamps to the L1-WCAP data field after rising edges (at pin CC6) and to the L2-WCAP data field after falling edges (at pin CC6):

```
L0 CNT  {reg=A, max=0x1FFFFFF}
L1 WCAP {reqnum=3, request=GENREQ, event=RISE, reg=A, pin=CC6}
L2 WCAP {reqnum=3, request=QUIET, event=FALL, reg=A, pin=CC7}
; HET HRSHARE feature configured to assign both WCAPs to pin CC6
```

The HTU frame will have two elements: The first gives the time stamp of the rising edge r(n) and the second gives the time stamp of the previous falling edge f(n-1). Using the code above, requests (R) and the quiet requests (QR) will occur at the times shown in Figure 18-12, and a request lost will only be signaled when the frame makes an access during the times marked with RL. So reading [22, 21] as frame elements is correct. If the signal frequency would increase, then a wrong pair [22, 23] could be read, but this will be signaled by a request lost error since at least e2 falls into the RL period.

Figure 18-12. Timing Example for Two WCAP Instructions



18.2.5 Memory Protection

This feature allows restricting accesses to certain areas in memory in order to protect critical application data from unintentionally being manipulated by the HTU.

If the HTU memory protection feature is disabled, the full 4 GB address range can be accessed by the HTU without exception. There are two memory regions that start and end addresses can be configured.

With the HTU memory protection feature enabled, read and write accesses by the HTU IFADDRA and IFADDRB registers inside the defined regions are allowed. HTU access to its tightly-coupled memory is independent of the MPU, it routes through the dedicated HTU/NHET bus using the IHADDR bits in the IHADDRCT register. See Section 18.2 for details on the tightly-coupled bus.

For accesses outside the regions, one of two modes is configurable:

- Any access performed by the HTU is forbidden and will be signaled to the ESM module. Write accesses will be blocked.
- Read access is allowed but write access will be blocked and signaled to the ESM module.

To use one region only, REG01ENA must be 0. Bits ACCR01, INTENA01, and register settings of MP1S and MP1E will be ignored.

To use both regions, the following rules must be followed:

1. Memory mapped region 0 covers a lower memory area as Memory mapped region 1.
2. REG01ENA is a 1 and REG0ENA is a 0.
3. ACCR01 is set for the desired access type, ACCR0 is ignored.
4. INTENA01 is set for the desired action, INTENA0 is ignored.

If an element transfer of DCP x generates a memory protection error, then:

1. The element counter of DCP x is cleared.
2. All new element transfers on DCP x are stopped.
3. The active busy bit of DCP x is cleared.
4. DCP x is disabled in the CPENA register. The DCPs other than DCP x will not be affected.
5. The FT flag will be set.
6. An error is signaled to the ESM module.

18.2.6 Control Packet RAM Parity Checking

The HTU module can detect parity errors in the DCP (Double Control Packet) RAM. DCP RAM parity checking is implemented using one parity bit per byte. Even or odd parity checking can be selected in the DEVCR1 register of the system module and can be enabled/disabled by a 4-bit key in the PCR register.

During a read access to the DCP RAM, the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. The parity check is performed when the HTU or any other master (for example, CPU) makes a read access to the DCP RAM. A read access within the RAM section of an initial or current DCP checks all 16 bytes of the DCP at a time (see also DCP memory map). For example, if a byte read access happens for DCP RAM address 0, but there is a parity error at byte address Ch then the parity error will occur and the captured parity address will be Ch and not 0. The address of the byte in which the error occurred can be read from the PAR register. If successive DCP RAM read accesses generate multiple parity errors, only the address of the first detected error will be captured and the PAR register will not be updated by subsequent errors until it is read by the application. When multiple errors in a 16 byte word are detected, only the address of the lowest byte will be captured.

The application can decide whether to stop any transfers when a parity error is detected or to continue transferring data. If the COPE (Continue On Parity Error) bit is 0 and parity checking is enabled, then the HTU will not start the frame and the corresponding DCP will be automatically disabled in the CPENA register. If a master other than the HTU (for example, CPU) reads the RAM section of DCP x and a parity error is detected during this read access, while the parity check is enabled and the COPE bit is 0, then the DCP x will be automatically disabled in the CPENA register. If a frame for this DCP x is ongoing during this read access, then in addition the element counter of DCP x is cleared, all new element transfers on DCP x are stopped and the active busy bit of DCP x is cleared. With COPE set to 1 and the parity check enabled, the parity checking will still be performed, but the data transfer of an active DCP continues after a parity error was detected for this DCP. So neither the DCP with the parity error will be disabled nor the frame will be stopped.

After a DCP is enabled (with CPENA using BIM = 0), then at the start of the first frame, the HTU performs the parity check only on the initial DCP, since it does not need the current DCP information. For further frames, the HTU performs the parity check for both initial and current DCP, since it needs both information.

On a parity error detection, an error will also be signaled to the ESM module.

18.2.6.1 Parity Bit Mapping and Testing

To test the parity checking mechanism, the parity RAM can be made accessible in order to allow manual fault insertion. Once the TEST bit is set, the parity bits are mapped to address FF4E 0200h.

When in test mode (the parity RAM is accessible), no parity checking will be done when reading from parity RAM, but parity checking will still be performed for read accesses to the DCP RAM.

Table 18-3 and Table 18-4 show how the corresponding parity bits of the DCP RAM bytes are mapped into the memory.

Each byte in DCP RAM has its own parity bit in the DCP Parity RAM. P0 is the parity bit for byte 0, P1 is the parity bit for byte 1 and so on.

Table 18-3. DCP RAM

Bit	31 24	23 16	15 8	7 0
FF4E 0000h	Byte 0	Byte 1	Byte 2	Byte 3
FF4E 0004h	Byte 4	Byte 5	Byte 6	Byte 7
FF4E 0008h	Byte 8	Byte 9	Byte 10	Byte 11
FF4E 000Ch	Byte 12	Byte 13	Byte 14	Byte 15

Table 18-4. DCP Parity RAM

Bit		24		16		8		0
FF4E 0200h		P0		P1		P2		P3
FF4E 0204h		P4		P5		P6		P7
FF4E 0208h		P8		P9		P10		P11
FF4E 020Ch		P12		P13		P14		P15

18.2.6.2 Initializing Parity Bits

After device power up, the DCP RAM content including the parity bit cannot be guaranteed. In order to avoid parity failures, when reading DCP RAM, the RAM has to be initialized first. This can simply be done by writing known values into the RAM by software and the corresponding parity bit will be automatically calculated.

Another possibility to initialize the DCP memory and its parity bits is to use the system module, which is an on-chip module external to the HTU. This module can start the automatic initialization of all RAMs on the microcontroller including the HTU DCP RAM. This function initializes the complete DCP RAM to "0' when activated by the system module. Depending on the even/odd parity selection, all parity bits will be calculated accordingly. The HTUEN bit must be cleared and the parity functionality must be enabled (by PARITY_ENA) during the automatic DCP RAM initialization. If HTUEN is 1 when the initialization is triggered by the system module, then the initialization will not be performed and the HTU operation is not affected. If a 1 is written to HTUEN during the initialization, then the HTUEN bit will be set but the HTU will not be enabled before the initialization completes.

18.3 Use Cases

18.3.1 Example: Single Element Transfer with One Trigger Request

This example considers the case that the HTU fills a RAM buffer in the main (CPU) data RAM. The HTU reads from the instruction which generates the HTU requests.

This example uses a PCNT instruction. Every time the PCNT has captured a new pulse or period value, it will automatically generate a transfer request to the HTU, which then transfers the value from the N2HET RAM to the buffer RAM. So over time consecutive locations in the RAM buffer can be filled with consecutive measurement values captured into the N2HET RAM data field of the same PCNT instruction without loading or interrupting the CPU.

18.3.2 Example: Multiple Element Transfer with One Trigger Request

The following example shows how the HTU could be used to fill a RAM buffer with a data stream including different types of measurement values belonging to the same N2HET input signal (on one pin): Time stamp values (WCAP), edge counter values (ECNT) and last period values (PCNT).

Figure 18-13 shows the timing and Table 18-5 shows the byte addresses of the program- (PF), control- (CF), data- (DF) and reserved field (res) of the WCAP-ECNT-PCNT instruction block. The timing and code example assumes that all three instructions are assigned to the same N2HET pin.

Figure 18-13. Timing of the WCAP, ECNT, PCNT Example

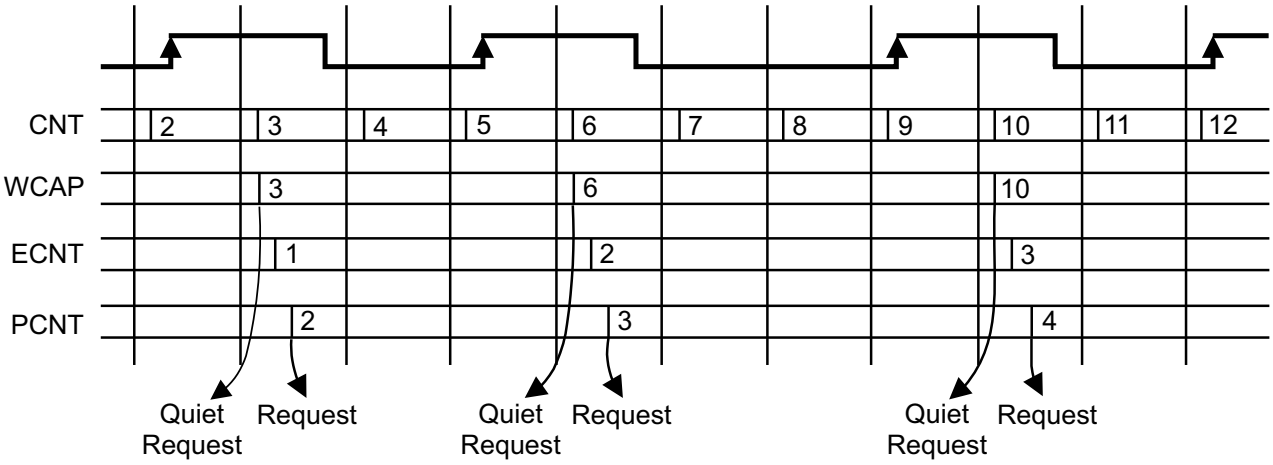


Table 18-5. Field Addresses of the WCAP, ECNT, PCNT Example

	PF	CF	DF	Res
WCAP	30h	34h	38h	3Ch
ECNT	40h	44h	48h	4Ch
PCNT	50h	54h	58h	5Ch

In the HET code, the HTU request is enabled only for the last instruction (PCNT) of the WCAP-ECNT-PCNT block. When the PCNT condition is true, it will cause the generated HTU frame to perform three HTU element reads from the data fields of WCAP, ECNT, and PCNT.

32-Bit-Transfer of data fields:

Table 18-6 shows how the internal element counter, frame counter, and the address registers change over time for the example described above. Every time the PCNT instruction captures a new value it generates a request to the HTU, which starts a frame. At the end of each frame the frame counter decrements.

Table 18-6. 32-Bit-Transfer of Data Fields (1)

Frame Counter	3			2			1		
Element Counter	3	2	1	3	2	1	3	2	1
Source Address (HET)	38h	48h	58h	38h	48h	58h	38h	48h	58h
Destination Address (main CPU RAM)	70h	74h	78h	7Ch	80h	84h	88h	8Ch	90h

(1) The diagram shows the byte addresses

The destination buffer is filled with the WCAP, ECNT, and PCNT data field values shown in Table 18-7.

Table 18-7. Destination Buffer Values

Address	Frame Count	Instruction	Value
70h	3	WCAP	3
74h	3	ECNT	1
78h	3	PCNT	2
7Ch	2	WCAP	6
80h	2	ECNT	2
84h	2	PCNT	3
88h	1	WCAP	10
8Ch	1	ECNT	3
90h	1	PCNT	4

The corresponding setup of the HTU control packet for this example is as follows:

```
IHADDR    = 0x38                // points to WCAP data field
IFADDRA   = 0x70                // points to buffer
ITCOUNT [frame count = 3] [element count = 3]
IHADDRCT = [DIR: Read HET and write to full address]
           [SIZE: 32 bit]
           [ADDMH: Increment HET address by 16 bytes]
           [ADDMF: Post increment full address mode]
           [Any transfer mode]
```

18.3.3 Example: 64-Bit-Transfer of Control Field and Data Fields

Table 18-8 shows how the internal element counter, frame counter, and the address registers change over time assuming the same example as in Section 18.3.2, but now with a transfer size set to 64-bit. The HET address now points to the control field of the instruction, so CF and DF are transferred as 64 bit data.

Table 18-8. 64-Bit-Transfer of Control Field and Data Fields (1)

Frame Counter	3			2			1		
Element Counter	3	2	1	3	2	1	3	2	1
HET (Source) Address	34h	44h	54h	34h	44h	54h	34h	44h	54h
Full (Destination) Address	70h	78h	80h	88h	90h	98h	A0h	A8h	B0h

(1) The diagram shows the byte addresses.

The destination buffer is filled with the WCAP, ECNT, and PCNT control and data field values shown in Table 18-9.

Table 18-9. Destination Buffer Values

Address	Frame Count	Instruction	Value
70h	3	WCAP	Control Field Value
74h	3	WCAP	3
78h	3	ECNT	Control Field Value
7Ch	3	ECNT	1
80h	3	PCNT	Control Field Value
84h	3	PCNT	2
88h	2	WCAP	Control Field Value
8Ch	2	WCAP	6
90h	2	ECNT	Control Field Value
94h	2	ECNT	2
98h	2	PCNT	Control Field Value
9Ch	2	PCNT	3
A0h	1	WCAP	Control Field Value
A4h	1	WCAP	10
A8h	1	ECNT	Control Field Value
ACh	1	ECNT	3
B0h	1	PCNT	Control Field Value
B4h	1	PCNT	4

The necessary setup of the HTU control packet (see Section 18.5) for this example is as follows:

```
IHADDR   = 0x34  (points to WCAP control field)
IFADDR   = 0x70  (points to buffer)
ITCOUNT [frame count = 3] [element count = 3]
IHADDRCT = [DIR: Read HET and write to full address]
           [SIZE: 64 bit]
           [ADDMH: Increment HET address by 16 bytes]
           [ADDMF: post increment full address mode]
           [Any transfer mode]
```

For different applications, which have the transfer direction set for reading the buffer and writing to HET fields, the 64-bit transfer could be used to change the conditional addresses together with a new data field.

18.4 Control Registers

Table 18-10 provides a summary of the registers. The registers support 8-bit, 16-bit, and 32-bit writes. The offset is relative to the associated peripheral select. See the following sections for detailed descriptions of the registers. The base address for the control registers is FFF7 A400h. The address locations not listed, are reserved.

Table 18-10. Control Register Mapping

Offset	Acronym	Register Description	Section
00h	HTU GC	Global Control Register	Section 18.4.1
04h	HTU CPENA	Control Packet Enable Register	Section 18.4.2
08h	HTU BUSY0	Control Packet Busy Register 0	Section 18.4.3
0Ch	HTU BUSY1	Control Packet Busy Register 1	Section 18.4.4
10h	HTU BUSY2	Control Packet Busy Register 2	Section 18.4.5
14h	HTU BUSY3	Control Packet Busy Register 3	Section 18.4.6
18h	HTU ACPE	Active Control Packet and Error Register	Section 18.4.7
20h	HTU RLBECTRL	Request Lost and Bus Error Control Register	Section 18.4.8
24h	HTU BFINTS	Buffer Full Interrupt Enable Set Register	Section 18.4.9
28h	HTU BFINTC	Buffer Full Interrupt Enable Clear Register	Section 18.4.10
2Ch	HTU INTMAP	Interrupt Mapping Register	Section 18.4.11
34h	HTU INTOFF0	Interrupt Offset Register 0	Section 18.4.12
38h	HTU INTOFF1	Interrupt Offset Register 1	Section 18.4.13
3Ch	HTU BIM	Buffer Initialization Mode Register	Section 18.4.14
40h	HTU RLOSTFL	Request Lost Flag Register	Section 18.4.15
44h	HTU BFINTFL	Buffer Full Interrupt Flag Register	Section 18.4.16
48h	HTU BERINTFL	BER Interrupt Flag Register	Section 18.4.17
4Ch	HTU MP1S	Memory Protection 1 Start Address Register	Section 18.4.18
50h	HTU MP1E	Memory Protection 1 End Address Register	Section 18.4.19
54h	HTU DCTRL	Debug Control Register	Section 18.4.20
58h	HTU WPR	Watch Point Register	Section 18.4.21
5Ch	HTU WMR	Watch Mask Register	Section 18.4.22
60h	HTU ID	Module Identification Register	Section 18.4.23
64h	HTU PCR	Parity Control Register	Section 18.4.24
68h	HTU PAR	Parity Address Register	Section 18.4.25
70h	HTU MPCS	Memory Protection Control and Status Register	Section 18.4.26
74h	HTU MP0S	Memory Protection 0 Start Address Register	Section 18.4.27
78h	HTU MP0E	Memory Protection 0 End Address Register	Section 18.4.28

18.4.1 Global Control Register (HTU GC)

Figure 18-14. Global Control Register (HTU GC) [offset = 00]

31	25	24	23	17	16
Reserved		VBUSHOLD	Reserved		HTUEN
R-0		R/WP-0	R-0		R/WP-0
15	9	8	7	1	0
Reserved		DEBM	Reserved		HTURES
R-0		R/WP-0	R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 18-11. Global Control Register (HTU GC) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	VBUSHOLD		Hold the VBUS bus.
		0	The VBUS is not held.
		1	The VBUSHOLD bit holds the bus used to transfer data between the HTU and the N2HET module. When the BUS_BUSY bit is 0, then the bus is no longer busy. While the bus is held, requests will still be accepted. They will be acted upon when the VBUSHOLD is 0. Request lost conditions will be detected and interrupts generated if they are enabled.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	HTUEN		Transfer Unit Enable Bit.
		0	The Transfer Unit is disabled.
		1	The Transfer Unit is enabled.
			The configuration registers and control packets should be set up first before the HTUEN bit is set to 1 to prevent it from carrying out unintended bus transactions. If the HTUEN bit is cleared to 0 during a frame is transferred, then the frame will be completed before the HTU is disabled.
			The HTUEN bit must be cleared and the parity functionality must be enabled (by PARITY_ENA) during the automatic DCP RAM initialization (see Initializing Parity Bits). If HTUEN is 1 when the initialization is triggered by the system module, then the initialization will not be performed and the HTU operation is not affected. If a 1 is written to HTUEN during the initialization, then the HTUEN bit will be set but the HTU will not be enabled before the initialization completes.
			Note: If HTU is disabled during a frame transfer, then the ongoing current frame will be completed before the HTU module is disabled. If enabled again, then the transfer will restart from the initial frame count for the CP programmed.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	DEBM		Debug Mode
		0	The Transfer Unit is stopped in debug mode.
			The HTU will complete the current frame, but not start any new frames. It will also ignore all requests from the HET and not generate any request lost signals.
		1	The Transfer Unit continues operation in debug mode.
			Note: Since the HET has also an "ignore suspend" bit, there a several possibilities for the behavior of the HET and HTU in suspend mode.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	HTURES		HTU Software Reset Request.
		0	Reset request is not issued to the HTU module. Writing a 0 has no effect.
		1	Reset request is issued to the HTU module.
			Ongoing element transfers will be completed, before resetting the complete HTU module, similar to a hardware reset. The HTURES bit will also be cleared. The recommended order of operations is: • Set the software reset bit. This also clears HTUEN. • Wait for the HTURES bit to clear. • Configure the HTU registers and packets. • Set the HTUEN bit to begin operation.

18.4.2 Control Packet Enable Register (HTU CPENA)

This register enables or disables the individual double control packets (DCP).

Figure 18-15. Control Packet Enable Register (HTU CPENA) [offset = 04h]

31	16
Reserved	
R-0	

15	0
CPENA	
R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 18-12. Control Packet Enable Register (HTU CPENA) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CPENA		CP Enable Bits Bits (2*x) and (2*x+1) of CPENA control the double control packet (DCP) x (whereby x must be within 0,1,....,7). See Table 18-13 for write rules. See Table 18-14 for read rules.

Table 18-13. CPENA Write Results

Bit (2*x+1)	Bit (2*x)	Control packets (CP) B and A of DCP x are affected as follows:
0	0	CP B and A are not changed.
0	1	CP B is disabled and CP A are enabled simultaneously.
1	0	CP B is enabled and CP A are disabled simultaneously.
1	1	CP B and CP A are both disabled simultaneously.

Table 18-14. CPENA Read Results

Bit (2*x+1)	Bit (2*x)	State of DCP:
0	0	The DCP is disabled.
0	1	CP B is disabled and CP A is enabled.
1	0	CP B is enabled and CP A is disabled.
1	1	Cannot be read.

- The conditions listed in Section 18.2.3 can automatically disable DCP x. In this case bits (2*x) and (2*x+1) are both automatically set to 0.
- When bits (2*x) and (2*x+1) change from 00 to 01 or from 00 to 10 caused by a write access to CPENA, then old pending requests on the corresponding request line are cleared. This means only new requests which occur **after** this write access cause the first HTU transfer for this DCP. This is **not** the case when **switching** CPs (from 10 to 01 or from 01 to 10).
- CP A and/or CP B of a DCP can be configured to one-shot, circular or auto-switch transfer mode via the TMBA or TMBB bits in the IHADDRCT control packet configuration. If a write access to CPENA occurs during the last frame of a buffer (with frame counter = 1) then the action defined by the write access to CPENA and the action defined by TMBx can contradict. The priority rules for this case are given in Table 18-1.

18.4.3 Control Packet (CP) Busy Register 0 (HTU BUSY0)

This register displays the status of individual control packets.

Figure 18-16. Control Packet (CP) Busy Register 0 (HTU BUSY0) [offset = 08h]

31	25	24	23	17	16
Reserved		BUSY0A	Reserved		BUSY0B
R-0		R/W1CP-0	R-0		R/W1CP-0

15	9	8	7	1	0
Reserved		BUSY1A	Reserved		BUSY1B
R-0		R/W1CP-0	R-0		R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; *-n* = value after reset

Table 18-15. Control Packet (CP) Busy Register 0 (HTU BUSY0) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUSY0A		Busy Flag for CP A of DCP 0.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUSY0B		Busy Flag for CP B of DCP 0.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY1A		Busy Flag for CP A of DCP 1.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	BUSY1B		Busy Flag for CP B of DCP 1.

The bit is **set** when the frame on the according control packet **starts** (there could be a delay between the request and the start of the frame).

The bit is automatically **cleared** at any of the following conditions:

1. At the end of a frame.
2. Writing a 1 to a BUSY bit (of DCP x) if that bit is 1. This will:
   a. clear the element counter of DCP x,
   b. stop all new element transfers on DCP x
   c. clear the busy bit and (4.) disable DCP x in the CPENA register.
   d. there is no effect if the BUSY bit is 0.
3. At the conditions listed in Section 18.2.3.

A write access to the CPENA register can stop a control packet (CP) in single buffer mode or it can switch to the other CP of a DCP in dual buffer mode. If stopping or switching occurs while a frame runs on the currently active control packet, the CPU can poll the busy bit to determine when it is safe to read the buffer.

18.4.4 Control Packet (CP) Busy Register 1 (HTU BUSY1)

This register displays the status of individual control packets.

Figure 18-17. Control Packet (CP) Busy Register 1 (HTU BUSY1) [offset = 0Ch]

31	25	24	23	17	16
Reserved		BUSY2A	Reserved		BUSY2B
R-0		R/W1CP-0	R-0		R/W1CP-0

15	9	8	7	1	0
Reserved		BUSY3A	Reserved		BUSY3B
R-0		R/W1CP-0	R-0		R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; -*n* = value after reset

Table 18-16. Control Packet (CP) Busy Register 1 (HTU BUSY1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUSY2A		Busy Flag for CP A of DCP 2.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUSY2B		Busy Flag for CP B of DCP 2.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY3A		Busy Flag for CP A of DCP 3.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	BUSY3B		Busy Flag for CP B of DCP 3.

See Section 18.4.3 for more details.

18.4.5 Control Packet (CP) Busy Register 2 (HTU BUSY2)

Figure 18-18. Control Packet (CP) Busy Register 2 (HTU BUSY2) [offset = 10h]

31	25	24	23	17	16
Reserved		BUSY4A	Reserved		BUSY4B
R-0		R/W1CP-0	R-0		R/W1CP-0

15	9	8	7	1	0
Reserved		BUSY5A	Reserved		BUSY5B
R-0		R/W1CP-0	R-0		R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; -*n* = value after reset

Table 18-17. Control Packet (CP) Busy Register 2 (HTU BUSY2) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUSY4A		Busy Flag for CP A of DCP 4.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUSY4B		Busy Flag for CP B of DCP 4.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY5A		Busy Flag for CP A of DCP 5.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	BUSY5B		Busy Flag for CP B of DCP 5.

18.4.6 Control Packet (CP) Busy Register 3 (HTU BUSY3)

Figure 18-19. Control Packet (CP) Busy Register 3 (HTU BUSY3) [offset = 14h]

31	25	24	23	17	16
Reserved		BUSY6A	Reserved		BUSY6B
R-0		R/W1CP-0	R-0		R/W1CP-0

15	9	8	7	1	0
Reserved		BUSY7A	Reserved		BUSY7B
R-0		R/W1CP-0	R-0		R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; *-n* = value after reset

Table 18-18. Control Packet (CP) Busy Register 3 (HTU BUSY3) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUSY6A		Busy Flag for CP A of DCP 6.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUSY6B		Busy Flag for CP B of DCP 6.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY7A		Busy Flag for CP A of DCP 7.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	BUSY7B		Busy Flag for CP B of DCP 7.

18.4.7 Active Control Packet and Error Register (HTU ACPE)

Figure 18-20. Active Control Packet and Error Register (HTU ACPE) [offset = 18h]

31	30	29	28	24	23	20	19	16
ERRF	Reserved		ERRETC		Reserved		ERRCPN	
R/W1CP-0	R-0		R-0		R-0		R-0	

15	14	13	12	8	7	4	3	0
TIPF	BUSBUSY	Rsvd	CETCOUNT		Reserved		NACP	
R-0	R-0	R-0	R-0		R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; *-n* = value after reset

Table 18-19. Active Control Packet and Error Register (HTU ACPE) Field Descriptions

Bit	Field	Value	Description
31	ERRF		Error Flag
		0	No error occurred.
		1	This bit is set when one of the conditions listed at ERRETC is fulfilled and ERRETC and ERRCPN are captured. Once ERRF is set, it is cleared when reading the upper 16-bit word of the ACPE register or the complete 32-bit register. It is also cleared when writing a 1 to ERRF. ERRF can be used to indicate if ERRETC and ERRCPN contain new unread data.
30-29	Reserved	0	Reads return 0. Writes have no effect.

Table 18-19. Active Control Packet and Error Register (HTU ACPE) Field Descriptions (continued)

Bit	Field	Value	Description
28-24	ERRETC		Error Element Transfer Count If one of the following conditions happens the current element transfer counter of the control packet (specified by ERRCPN) is captured to ERRETC. Please see Section 18.2.3. • Request Lost Error of control packet specified by ERRCPN. This is independent of the CORL bit. • Parity Error of control packet specified by ERRCPN. This requires the parity check to be enabled, but is independent of the COPE bit. • Bus Error of control packet specified by ERRCPN. • Memory Protection Error of control packet specified by ERRCPN. This requires the memory protection to be enabled. • Writing a 1 to a BUSY bit, which belongs to the control packet specified by ERRCPN, if that bit is 1. There is no effect, if the BUSY bit is 0. ERRETC is frozen from being updated until the upper 16-bit word of the ACPE register or the complete 32-bit register is read by the CPU. After this read, the HTU will update ERRETC if one of the above conditions is fulfilled again. During debugging, ERRETC stays frozen even when reading the upper 16-bit word or the 32-bit register.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	ERRCPN		Error Control Packet Number If one of the conditions listed at ERRETC happens the number of the control packet, which caused the condition, is captured to ERRCPN. **Control Packet** — **ERRCPN Value** CP A of DCP x — 2 x CP B of DCP x — 2 x+1 With x = 0,1,...or 7 ERRCPN is frozen from being updated until the upper 16-bit word of the ACPE register or the complete 32-bit register is read by the CPU. After this read, the HTU will update ERRCPN if one of the above conditions is fulfilled again. During debugging, ERRCPN stays frozen even when reading the upper 16-bit word or the 32-bit register. If one of the conditions is fulfilled ERRETC and ERRCPN are updated simultaneously.
15	TIPF		Transfer in Progress Flag
		0	No transfers are in progress.
		1	A transfer is currently active. This bit is the result of a logical OR function of all BUSYxx flags of the 4 BUSYx registers.
14	BUSBUSY		Bus is Busy
		0	Bus between N2HET and HTU is not busy.
		1	When BUSBUSY is 1, the bus is busy with a transfer. It is different from TIPF above because BUSBUSY will go low after VBUSHOLD is set to 1 and no transfers are pending between the HTU and the main memory. TIPF will remain 1, if a transfer is still pending and VBUSPHOLD is 1.
13	Reserved	0	Reads return 0. Writes have no effect.
12-8	CETCOUNT		Current Element Transfer Count CETCOUNT shows the current element transfer counter for the frame that is currently processed. If the HTU does not currently transfer any frame, CETCOUNT is 0. CETCOUNT is updated after the write part of a transfer. There is a period of up to 7 cycles between the time the CETCOUNT is 0 and the HTU is finished updating the current DCP (and the CPENA registers if the required conditions are fulfilled).
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	NACP		Number of Active Control Packet Indicates which CP currently processes a frame. **Active or Recent DCP** — **NACP Value** CP A of DCP x — 2 x CP B of DCP x — 2 x+1 With x = 0,1,...or 7 NACP is updated at the time the frame starts on the according CP, and it is updated with a new value when a frame starts on a different CP. Note, that there can be a delay between the request and the start of the frame.

18.4.8 Request Lost and Bus Error Control Register (HTU RLBECTRL)

Figure 18-21. Request Lost and Bus Error Control Register (HTU RLBECTRL) [offset = 20h]

31	17	16
Reserved		BERINTENA
R-0		R/WP-0

15	9	8	7	1	0
Reserved		CORL	Reserved		RLINTENA
R-0		R/WP-0	R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

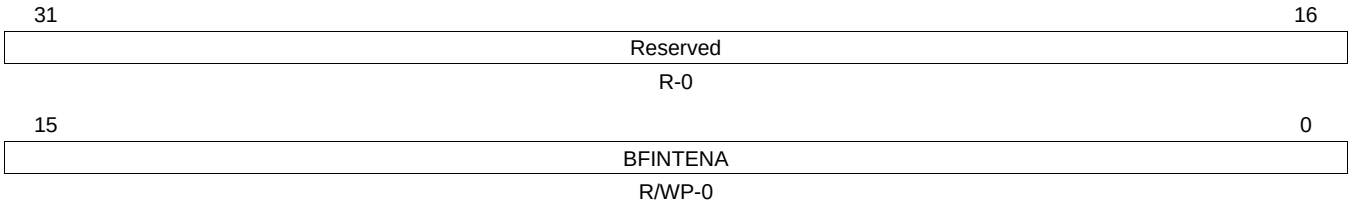
Table 18-20. Request Lost and Bus Error Control Register (HTU RLBECTRL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	BERINTENA		Bus Error Interrupt Enable Bit.
		0	The bus error interrupt is disabled for all DCPs.
		1	The bus error interrupt is enabled for all DCPs.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	CORL		Continue On Request Lost Error.
		0	Stop current frame on request lost detection. Please see Conditions for Frame Transfer Interruption.
		1	If CORL is 1 and DCP x is enabled, then DCP x will stay enabled after a request lost condition on DCP x and element transfers will continue.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	RLINTENA		Request Lost Interrupt Enable Bit.
		0	The request lost interrupt is disabled for all DCPs. Disabling RLINTENA will not clear the flags in the RLOSTFL register.
		1	The request lost interrupt is enabled for all DCPs. If bits are set in the RLOSTFL flag register at the time RLINTENA is (re-) enabled, then the according interrupt(s) will occur (in the order of the priority of the request lines).

18.4.9 Buffer Full Interrupt Enable Set Register (HTU BFINTS)

This registers allows to enable the buffer full interrupts for the different control packets. Reading registers BFINTS and BFINTC will return the same bits indicating the status which interrupt is enabled (1) or disabled (0).

Figure 18-22. Buffer Full Interrupt Enable Set Register (HTU BFINTS) [offset = 24h]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

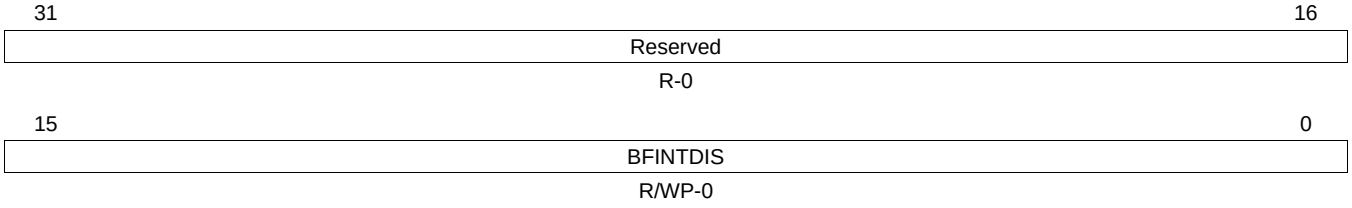
Table 18-21. Buffer Full Interrupt Enable Set Register (HTU BFINTS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BFINTENA		Bus Full Interrupt Enable Bits. If the interrupt for CP A of a DCP is enabled, then the interrupt is generated once buffer A is full, that is, once the frame counter CFTCTA decrements to 0. The same applies for CP B (and CFTCTB).
		0	Interrupt is disabled. Writing a 0 has no effect.
		1	Writing to bit (2*x) enables the interrupt for CP A of DCP x.
			Writing to bit (2*x+1) enables the interrupt for CP B of DCP x.

18.4.10 Buffer Full Interrupt Enable Clear Register (HTU BFINTC)

This registers allows to disable the buffer full interrupts for the different control packets. Reading registers BFINTS and BFINTC will return the same bits indicating the status which interrupt is enabled (1) or disabled (0)

Figure 18-23. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) [offset = 28h]



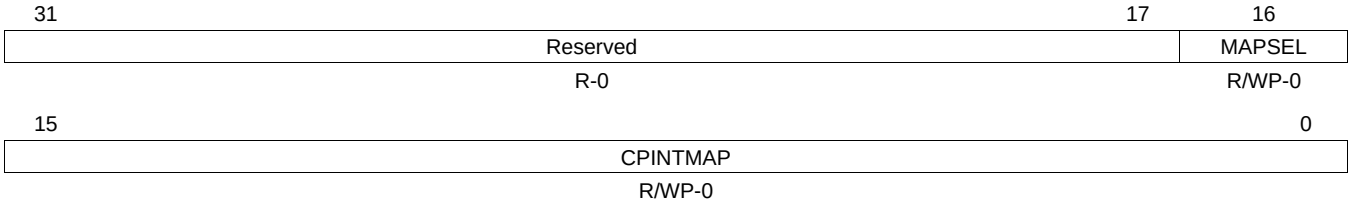
LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 18-22. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BFINTDIS		Buffer Full Interrupt Disable Bits.
		0	Interrupt is disabled. Writing a 0 has no effect.
		1	Writing to bit (2*x) disables the interrupt for CP A of DCP x.
			Writing to bit (2*x+1) disables the interrupt for CP B of DCP x.

18.4.11 Interrupt Mapping Register (HTU INTMAP)

Figure 18-24. Interrupt Mapping Register (HTU INTMAP) [offset = 2Ch]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 18-23. Interrupt Mapping Register (HTU INTMAP) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	MAPSEL		Interrupt Mapping Select Bit.
		0	If MAPSEL is 0, then one bit of CPINTMAP selects one of two interrupt priorities 0 or 1 for the **buffer full** interrupt for the according CP. The **request lost** and **bus error** interrupts of all CPs are set to priority 0, independent of CPINTMAP.
		1	If MAPSEL is 1, then one bit of CPINTMAP determines if the **buffer full**, **request lost** and **bus error** interrupts of the according CP are assigned either to interrupt line 0 or to 1.
15-0	CPINTMAP		CP Interrupt Mapping Bits.
		0	Interrupt of CP A (bit 2-x) of DCP x is mapped to interrupt line 0. Interrupt of CP B (bit 2*x+1) of DCP x is mapped to interrupt line 0.
		1	Interrupt of CP A (bit 2-x) of DCP x is mapped to interrupt line 1. Interrupt of CP B (bit 2*x+1) of DCP x is mapped to interrupt line 1.

18.4.12 Interrupt Offset Register 0 (HTU INTOFF0)

The INTOFF0 register reflects the highest priority interrupt flag bit set in the BERINTFL, RLOSTFL, or BFINTFL flag registers with the appropriate CPINTMAP bit set to 0. The priority order (from high to low) is: BER, RLOST, buffer-full. Interrupts for request lines with lower number have higher priority.

Figure 18-25. Interrupt Offset Register 0 (HTU INTOFF0) [offset = 34h]

31	16
Reserved	
R-0	

15	10	9	8	7	4	3	0
Reserved		INTTYPE0		Reserved		CPOFF0	
R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 18-24. Interrupt Offset Register 0 (HTU INTOFF0) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	INTTYPE0		Interrupt Type of Interrupt Line 0. Indicates whether a buffer-full, RLOST or BER interrupt, assigned to interrupt line 0, is currently pending.
		0	No interrupt.
		1h	Interrupt is caused by full buffer on CP/DCP specified by CPOFF0.
		2h	RLOST interrupt is generated by CP/DCP specified by CPOFF0.
		3h	BER interrupt is generated by CP/DCP specified by bits CPOFF0.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	CPOFF0		CP Offset. Indicates for which control packet the interrupt is pending, which is classified by INTTYPE0 and is assigned to interrupt line 0.
		0	DCP 0, CP A
		1h	DCP 0, CP B
		2h	DCP 1, CP A
		3h	DCP 1, CP B
		4h	DCP 2, CP A
		5h	DCP 2, CP B
		6h	DCP 3, CP A
		7h	DCP 3, CP B
		8h	DCP 4, CP A
		9h	DCP 4, CP B
		Ah	DCP 5, CP A
		Bh	DCP 5, CP B
		Ch	DCP 6, CP A
		Dh	DCP 6, CP B
		Eh	DCP 7, CP A
		Fh	DCP 7, CP B

NOTE: Reading CPOFF0 will clear the bit generating the current interrupt from appropriate flag register (BERINTFL, RLOSTFL, or BFINTFL), except when in debug mode where reading CPOFF0 will have no effect on the flag registers.

In order to read INTTYPE0 and CPOFF0 simultaneously, always read this register using word or half-word but not using byte accesses.

18.4.13 Interrupt Offset Register 1 (HTU INTOFF1)

This register is organized identically to the INTOFF0 register. The difference is that INTOFF1 reflects the highest priority interrupt flag bit set in the BERINTFL, RLOSTFL, or BFINTFL flag registers with the appropriate CPINTMAP bit set to 1.

Figure 18-26. Interrupt Offset Register 1 (HTU INTOFF1) [offset = 38h]

31	16
Reserved	
R-0	

15	10	9	8	7	4	3	1	0
Reserved		INTTYPE1		Reserved		CPOFF1		
R-0		R-0		R-0		R-0		

LEGEND: R = Read only; -*n* = value after reset

Table 18-25. Interrupt Offset Register 1 (HTU INTOFF1) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	INTTYPE1		Interrupt Type of Interrupt Line 1. Indicates whether a buffer-full, RLOST or BER interrup, assigned to interrupt line 1, is currently pending.
		0	No interrupt.
		1h	Interrupt is caused by full buffer on CP/DCP specified by CPOFF1.
		2h	RLOST interrupt is generated by CP/DCP specified by CPOFF1.
		3h	BER interrupt is generated by CP/DCP specified by bits CPOFF1.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	CPOFF1		CP Offset. Indicates for which DCP/CP the interrupt is pending, which is classified by INTTYPE1 and is assigned to interrupt line 1
		0	DCP 0, CP A
		1h	DCP 0, CP B
		2h	DCP 1, CP A
		3h	DCP 1, CP B
		4h	DCP 2, CP A
		5h	DCP 2, CP B
		6h	DCP 3, CP A
		7h	DCP 3, CP B
		8h	DCP 4, CP A
		9h	DCP 4, CP B
		Ah	DCP 5, CP A
		Bh	DCP 5, CP B
		Ch	DCP 6, CP A
		Dh	DCP 6, CP B
		Eh	DCP 7, CP A
		Fh	DCP 7, CP B

NOTE: Reading CPOFF1 will clear the bit generating the current interrupt from appropriate flag register (BERINTFL, RLOSTFL, or BFINTFL), except when in debug mode where reading CPOFF1 will have no effect on the flag registers.

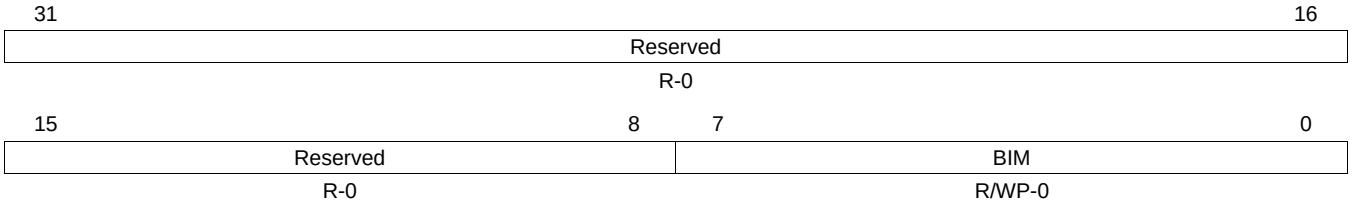
In order to read INTTYPE1 and CPOFF1 simultaneously, always read this register using word or half-word but not using byte accesses.

18.4.14 Buffer Initialization Mode Register (HTU BIM)

This register enables special applications, where one CP is temporarily disabled, but after having re-enabled the CP, filling the buffer should not start back at its beginning, but should continue after the last element of the previous run.

Table 18-27 shows more details on the BIM usage.

Figure 18-27. Buffer Initialization Mode Register (HTU BIM) [offset = 3Ch]



LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 18-26. Buffer Initialization Mode Register (HTU BIM) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	BIM		Buffer Initialization Mode The BIM bits and the TMBx bits determine when a buffer is initialized, that means when its initial full address IFADDRx and its initial frame counter IFTCOUNT is used. When initializing (restarting) a buffer the information in the corresponding initial DCP RAM is loaded to a internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx). The current DCP RAM is updated the first time when the first frame has finished. A buffer is initialized: • In circular buffer transfer mode (defined by TMBx) when the end of the buffer is reached. • When CPs are switched or enabled according to Buffer Initialization. The CPENA bits (2*x+1) and (2*x) are changed by write access to CPENA. For the first two rows of the table the change of the CPENA bits could also be the result of the auto switch feature (as defined by TMBx). BIM bit x only affects DCP x (with x = 0,1,...or 7).

Table 18-27. Buffer Initialization

Case	Change of CPENA Bits (2*x+1) and (2*x)		Action on Buffer A or B (of DCP x)		
	Old State [1]	New State [2]		BIM Bit x = 0 (normal mode)	BIM Bit x = 1 (special mode)
A	01	10	Switch from CP A to B	Next frame starts at the initial address of buffer B [3]	Same as for BIM bit x = 0
B	10	01	Switch from CP B to A	Next frame starts at the initial address of buffer A [3]	Same as for BIM bit x = 0
C	01	01	Stay at CP A	Write to CPENA bits (2*x+1) and (2*x) is ignored	Same as for BIM bit x = 0
E	10	10	Stay at CP B	Write to CPENA bits (2*x+1) and (2*x) is ignored	Same as for BIM bit x = 0
E	00	01	Enable CP A	Next frame starts at the **initial** address of buffer A	*Next frame continues at the **current** address of buffer A*
F	00	10	Enable CP B	Next frame starts at the **initial** address of buffer B	*Next frame continues at the **current** address of buffer B*
G	xx	11	Disable both CPs	Stop DCP x	Same as for BIM bit x = 0

(1) See read table of CPENA register (Table 18-14).
(2) See write table of CPENA register (Table 18-13).
(3) This is regardless of whether the switch is done by a write access to CPENA or by the auto switch feature.

NOTE: For cases E and F above, after the last frame of a buffer, the HTU sets CFTCTx to 0 and CFADDRx to the next address after the buffer. If the DCP was disabled during this state, then both CFTCTx and CFADDRx would contain invalid initialization values. Therefore, if a DCP should continue at its current address, then the software should use one of the following two procedures before it (re-) enables the DCP (as per Table 18-27):

1. `If CFTCTx ≠ 0 then set BIM=1`
   `If CFTCTx = 0 then set BIM=0`
2. `If CFTCTx ≠ 0 then set BIM=1`
   `If CFTCTx = 0 then {set BIM=1;`

   `set CFTCTx = IFTCOUNT;`
   `set CFADDRx = IFADDRx}`

But note that these procedures are only required for the cases E and F and not for all the other cases shown in Table 18-27. Also, when a buffer reaches its end in circular mode, it uses the initial DCP information to restart independently of the BIM setting (assuming it is not temporarily disabled during CFTCTx = 0).

NOTE: Similarly, care needs to be taken when BIM is set to 1 and a DCP is enabled for the very first time. Also, in this case, CFTCTx and CFADDRx usually contain invalid initialization values. The software can either solve this by setting BIM = 0 for the first time or setting CFADDRx to IFADDRx and CFTCTx to IFTCOUNT before the DCP is enabled.

NOTE: If

- the HTUEN bit is changed to 1 after the HTU was disabled HTUEN = 0
- the CPENA bit pair is 01 or 10 (during this HTUEN change)

then the corresponding BIM bit will decide if the corresponding buffer continues at its initial or current address. Cases E and F in Table 18-27 also apply for this situation. The software should use the procedures explained in the first note before setting HTUEN.

18.4.15 Request Lost Flag Register (HTU RLOSTFL)

Figure 18-28. Request Lost Flag Register (HTU RLOSTFL) [offset = 40h]

31	16
Reserved	
R-0	
15	0
CPRLFL	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 18-28. Request Lost Flag Register (HTU RLOSTFL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CPRLFL		CP Request Lost Flags
		0	No request was lost. Writing a 0 has no effect.
		1	If bit (2*x) is set, a request was lost on CP A of DCP x. If bit (2*x+1) is set, a request was lost on CP B of DCP x. Reading from INTOFFx in case of a RLOST interrupt clears the corresponding flag. The state of the flag bit can be polled even if RLINTENA is cleared. • Reading CPRLFL will not clear the flags or • Reading from INTOFFx clears the corresponding flag. • Writing a 1 clears the corresponding flag.

18.4.16 Buffer Full Interrupt Flag Register (HTU BFINTFL)

Figure 18-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) [offset = 44h]

31	16
Reserved	
R-0	
15	0
BFINTFL	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

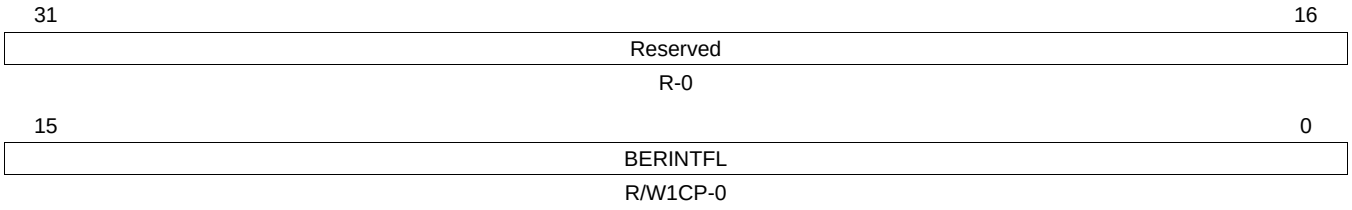
Table 18-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BFINTFL		Buffer Full Interrupt Flags
		0	No buffer full condition is detected. Writing a 0 has no effect.
		1	If bit (2*x) is set, a buffer full condition on CP A of DCP x has been detected. If bit (2*x+1) is set, a buffer full condition on CP B of DCP x has been detected. The BFINTFL flag is set after the last frame finishes on the corresponding buffer regardless of whether the buffer is configured to one shot, circular or auto-switch mode. If BFINTFL is set in circular mode, then a circular overrun has occurred on the corresponding buffer. This can be used to indicate whether the buffer section after the frozen full address contains valid data or not. Reading from INTOFFx in case of a buffer-full interrupt clears the corresponding flag. The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared. • Reading BFINTFL will not clear the flags or • Reading INTOFFx will clear the corresponding flags or • Writing a 1 clears the corresponding flag.

18.4.17 BER Interrupt Flag Register (HTU BERINTFL)

A bus error interrupt results due to an address error or a timeout condition on the main memory access. A bus error will stop the frame transfer. Please see Section 18.2.3.

Figure 18-30. BER Interrupt Flag Register (HTU BERINTFL) [offset = 48h]



LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 18-30. BER Interrupt Flag Register (HTU BERINTFL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BERINTFL		Bus Error Interrupt Flags
		0	No bus error condition is detected. Writing a 0 has no effect.
		1	If bit (2*x) is set, then a BER interrupt is pending on CP A of DCP x. If bit (2*x+1) is set, then a BER interrupt is pending on CP B of DCP x. The state of the flag bit can be polled even if BERINTENA is cleared. • Reading BERINTFL will not clear the flags or • Reading from INTOFFx in case of a BER interrupt clears the corresponding flag or • Writing a 1 clears the corresponding flag.

18.4.18 Memory Protection 1 Start Address Register (HTU MP1S)

This register configures the start address of memory protection region 1.

Figure 18-31. Memory Protection 1 Start Address Register (HTU MP1S) [offset = 4Ch]

31	16
STARTADDRESS1	
R/WP-0	

15	2	1	0
STARTADDRESS1		0	0
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 18-31. Memory Protection 1 Start Address Register (HTU MP1S) Field Descriptions

Bit	Field	Description
31-0	STARTADDRESS1	The start address defines at which main memory address the region begins. A memory protection error will be triggered, if the HTU accesses an address smaller than STARTADDRESS1 and the MPCS bit REG01ENA register is configured accordingly. The address is 32-bit aligned, so the 2 LSBs are not significant and will always read 0.

18.4.19 Memory Protection 1 End Address Register (HTU MP1E)

Figure 18-32. Memory Protection 1 End Address Register (HTU MP1E) [offset = 50h]

31	16
ENDADDRESS1	
R/WP-0	

15	2	1	0
ENDADDRESS1		0	0
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 18-32. Memory Protection 1 End Address Register (HTU MP1E) Field Descriptions

Bit	Field	Description
31-0	ENDADDRESS1	The end address defines at which address the region ends. A memory protection error will be triggered, if the HTU accesses an address bigger than ENDADDRESS1 and the register bit REG01ENA is configured accordingly. The address is 32-bit aligned, so the 2 LSBs are not significant and will always read 0. The effective end address is rounded up to the nearest word end address, that is, 0x200 = 0x203.

18.4.20 Debug Control Register (HTU DCTRL)

This register allows to create watch points on access to a certain location. It is intended to help debug the application execution during program development.

Figure 18-33. Debug Control Register (HTU DCTRL) [offset = 54h]

31	28	27	24	23	17	16
Reserved		CPNUM		Reserved		HTUDBGS
R-0		R-0		R-0		R/WS-0

15	1	0
Reserved		DBREN
R-0		R/WS-0

LEGEND: R/W = Read/Write; R = Read only; WS = Write in suspend mode only; *-n* = value after reset

Table 18-33. Debug Control Register (HTU DCTRL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	CPNUM		CP Number. These bit fields indicate the CP which should cause the watch point to match.
		0	CP A of DCP0
		1h	CP B of DCP0
		2h	CP A of DCP1
		3h	CP B of DCP1
		4h	CP A of DCP2
		5h	CP B of DCP2
		6h	CP A of DCP3
		7h	CP B of DCP3
		8h	CP A of DCP4
		9h	CP B of DCP4
		Ah	CP A of DCP5
		Bh	CP B of DCP5
		Ch	CP A of DCP6
		Dh	CP B of DCP6
		Eh	CP A of DCP7
		Fh	CP B of DCP7
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	HTUDBGS		HTU Debug Status. When the main memory address is equal to the unique address defined by WPR, or lies in the specified range resulting from WMR, then the HTUDBGS is set. If in addition DBREN is set, then the application code execution will be stopped. A 1 must be written to this bit in order to clear it and to release the CPU from debug halting state.
		0	Read: No watch point condition was detected. Write: No effect.
		1	Read: A watch point condition was detected. Write: Clears the bit.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	DBREN		Debug Request Enable If a watch point matches and DBREN is set, then the application code execution will be stopped. This bit can only be set or cleared when in debug mode. This bit and all other bits of the DCTRL, WPR and WMR registers are reset by the test reset (nTRST) but not by the normal device reset.

18.4.21 Watch Point Register (HTU WPR)

This register defines the main memory address of the watch point.

Figure 18-34. Watch Point Register (HTU WPR) [offset = 58h]

31	16
WP	
R/WS-0	

15	0
WP	
R/WS-0	

LEGEND: R/W = Read/Write; WS = Write in suspend mode only; -*n* = value after reset

Table 18-34. Watch Point Register (HTU WPR) Field Descriptions

Bit	Field	Description
31-0	WP	Watch Point Register A 32-bit address can be programmed into this register as a watch point. The WPR register is used along with the Watch Mask Register (WMR). When the main memory address is equal to the unique address defined by WPR, or lies in the specified range resulting from WMR, then the HTUDBGS is set. If in addition DBREN is set, then the application code execution is stopped. This register can only be programmed during debug mode. This register and all other bits of the DCTRL and WMR registers are reset by the test reset (nTRST) but not by the normal device reset.

18.4.22 Watch Mask Register (HTU WMR)

This register defines a mask of the main memory address of the watch point. It can be used to define a memory range in conjunction with the WPR register.

Figure 18-35. Watch Mask Register (HTU WMR) [offset = 5Ch]

31	16
WM	
R/WS-0	

15	0
WM	
R/WS-0	

LEGEND: R/W = Read/Write; WS = Write in suspend mode only; -*n* = value after reset

Table 18-35. Watch Mask Register (HTU WMR) Field Descriptions

Bit	Field	Description
31-0	WM	Watch Mask Register Setting a bit in the WMR register to 1 has the effect of masking the corresponding bit in of the main memory address, so that this bit is ignored for the address comparison. This register can only be programmed during debug mode. This register and all other bits of the DCTRL and WPR registers are reset by the test reset (nTRST) but not by the normal device reset.

18.4.23 Module Identification Register (HTU ID)

This register is for TI internal purposes and allows to keep track of the HTU module version on different devices.

Figure 18-36. Module Identification Register (HTU ID) [offset = 60h]

31	24	23	16
Reserved		CLASS	
R-0		R - Module Class Number	

15	8	7	0
TYPE		REV	
R - Class Subtype Number		R - Module Revision Number	

LEGEND: R = Read only; -*n* = value after reset

Table 18-36. Module Identification Register (HTU ID) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	CLASS		Module Class This field defines the module class number as read-only constant value for the HTU module. Writes have no effect.
15-8	TYPE		Subtype within a Class This field defines the subtype within a class as read-only constant value for the HTU module. Writes have no effect.
7-0	REV		Module Revision Number This field defines the module revision number as read-only constant value for the HTU module. Writes have no effect.

18.4.24 Parity Control Register (HTU PCR)

Figure 18-37. Parity Control Register (HTU PCR) [offset = 64h]

31																17	16
Reserved																	COPE
R-0																	R/WP-0
15							9	8	7			4	3				0
Reserved								TEST	Reserved				PARITY_ENA				
R-0								R/WP-0	R-0				R/WP-5h				

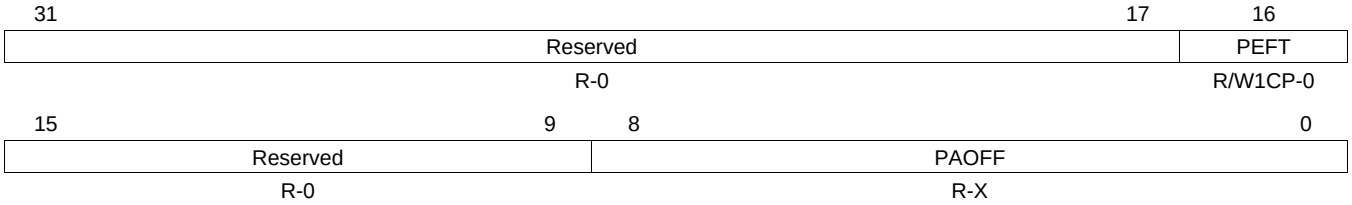
LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 18-37. Parity Control Register (HTU PCR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	COPE		Continue on Parity Error
		0	The HTU performs parity checks every time it reads the RAM section of DCP x (with x = 0, 1,... or 7), before the next frame (of DCP x) is started. If a parity error is detected during this read access and if the parity check is enabled, then the frame will not be started and DCP x will be automatically disabled in the CPENA register. If a master different than the HTU (for example, CPU) reads the RAM section of DCP x and a parity error is detected during this read access, while the parity check is enabled, then the DCP x will automatically be disabled in the CPENA register. If a frame is active on DCP x during this read access, then in addition the element counter of DCP x is cleared and all new element transfers on DCP x are stopped and the active busy bit of DCP x is cleared.
		1	The difference to COPE = 0 is, that the data transfer on a active DCP continues after a parity error was detected on this DCP. So neither the DCP with the parity error will be disabled nor the frame will be stopped.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	TEST		Test. When this bit is set, the parity bits are mapped into the peripheral RAM frame to make them accessible by the CPU.
		0	Parity bits are not memory-mapped.
		1	Parity bits are memory-mapped.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PARITY_ENA		Enable/Disable Parity Checking. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected, then the PEFT flag is set, PAOFF is captured if it is not currently frozen and an interrupt is generated if it is enabled.
		5h	Parity check is disabled.
		All Others	Parity check is enabled.
			Note: It is recommended to write Ah to enable error detection, to guard against single bit changes from flipping PARITY_ENA to a disable state.

18.4.25 Parity Address Register (HTU PAR)

Figure 18-38. Parity Address Register (HTU PAR) [offset = 68h]



LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; X = value is undefined; -*n* = value after reset

Table 18-38. Parity Address Register (HTU PAR) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
16	PEFT		Parity Error Fault Flag. This bit is set, when the HTU detects a parity error and parity checking is enabled.
		0	No fault is detected.
		1	Fault is detected.
			Note: Once PEFT is set, a read access to the lower 16 bits or to the complete 32-bit HTUPAR register will clear the PEFT flag in non-debug mode. Another possibility to clear PEFT is to write a 1 to the PEFT bit.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	PAOFF		Parity Error Address Offset. This bit field holds the address of the first parity error, which is detected in the DCP RAM. PAOFF provides the offset address of the erroneous byte counted from the beginning of the DCP memory. This error address is frozen from being updated until a read access to the lower 16 bits or to the complete 32-bit HTUPAR register happens. During debug mode, this address is frozen even when read.
			Note: The Parity Error Address bits will not be reset, neither by PORRST nor by any other reset source.

18.4.26 Memory Protection Control and Status Register (HTU MPCS)

Figure 18-39. Memory Protection Control and Status Register (HTU MPCS) [offset = 70h]

31	28	27	24
Reserved		CPNUM0	
R-0		R-0	

23	18	17	16
Reserved		MPEFT1	MPEFT0
R-0		R/W1CP-0	R/W1CP-0

15	12	11	8
Reserved		CPNUM1	
R-0		R-0	

7	6	5	4	3	2	1	0
Reserved		INT ENA01	ACCR01	REG01ENA	INT ENA0	ACCR0	REG0ENA
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 18-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	CPNUM0		Control Packet Number for single memory protection region configuration. CPNUM0 holds the number of the CP, which has caused the first memory protection error when only one memory protection region is used. This number is not updated for multiple access violations until it is read by the CPU. During debug mode, CPNUM0 is frozen even when read.
		0	CP A of DCP0
		1h	CP B of DCP0
		2h	CP A of DCP1
		3h	CP B of DCP1
		4h	CP A of DCP2
		5h	CP B of DCP2
		6h	CP A of DCP3
		7h	CP B of DCP3
		8h	CP A of DCP4
		9h	CP B of DCP4
		Ah	CP A of DCP5
		Bh	CP B of DCP5
		Ch	CP A of DCP6
		Dh	CP B of DCP6
		Eh	CP A of DCP7
		Fh	CP B of DCP7
23-18	Reserved	0	Reads return 0. Writes have no effect.
17	MPEFT1		Memory Protection Error Fault Flag 1. This bit is set, when the HTU performs an access outside the region defined by the MP0S and MP0E and the MP1S and MP1E registers, when the access violates the rights defined by ACCR01 and when the REG01ENA bit is set.
		0	No fault is detected. Writing a 0 has no effect.
		1	Fault is detected. Writing a 1 clears the bit.
16	MPEFT0		Memory Protection Error Fault Flag 0. This bit is set, when the HTU performs an access outside the region defined by the MP0S and MP0E registers, when the access violates the rights defined by ACCR and when the REG0ENA bit is set.
		0	No fault is detected. Writing a 0 has no effect.
		1	Fault is detected. Writing a 1 clears the bit.

Table 18-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions (continued)

Bit	Field	Value	Description
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	CPNUM1		Control Packet Number for single memory protection region configuration. CPNUM1 holds the number of the CP, which has caused the first memory protection error when only one memory protection region is used. This number is not updated for multiple access violations until it is read by the CPU. During debug mode, CPNUM1 is frozen even when read.
		0	CP A of DCP0
		1h	CP B of DCP0
		2h	CP A of DCP1
		3h	CP B of DCP1
		4h	CP A of DCP2
		5h	CP B of DCP2
		6h	CP A of DCP3
		7h	CP B of DCP3
		8h	CP A of DCP4
		9h	CP B of DCP4
		Ah	CP A of DCP5
		Bh	CP B of DCP5
		Ch	CP A of DCP6
		Dh	CP B of DCP6
		Eh	CP A of DCP7
		Fh	CP B of DCP7
7-6	Reserved	0	Reads return 0. Writes have no effect.
5	INTENA01		Interrupt Enable 01. This bit needs to be set when working with two memory mapped regions and a error should be generated to the ESM module on an access violation.
		0	Error signaling is disabled.
		1	Error signaling is enabled.
4	ACCR01		Access Rights 01. This bit defines the access rights for the HTU for accesses outside the region defined by the MP0S and MP0E and the MP1S and MP1E registers.
		0	HTU read access is allowed but write access will be signaled.
		1	Any access performed by the HTU is forbidden and will be signaled.
3	REG01ENA		Region Enable 01. This bit needs to be set when working with two memory-mapped regions. **REG0ENA must be set to 0** if this bit is set to a 1. Memory region 0 must be less than memory region 1.
		0	The protection outside the memory region defined by the MP0S and MP0E and the MP1S and MP1E registers is not enabled. This means the HTU can access any implemented memory space. REG0ENA could still enabled to give protection outside the MP0S : MP0E region.
		1	The protection outside the memory region defined by the MP0S and MP0E and the MP1S and MP0E registers is enabled. This means the HTU can perform any access within the regions, but if it attempts to perform a forbidden access outside of both of the regions (according to the ACCR01 configuration), the access is signaled by the MPEFT1 flag. The number of the CP, which has caused the memory protection error, is captured to CPNUM1 if it is not currently frozen and an error is generated if it is enabled.
2	INTENA0		Interrupt Enable 0. This bit needs to be set when working with one memory-mapped region and a error should be generated to the ESM module on an access violation.
		0	Error signaling is disabled.
		1	Error signaling is enabled.
1	ACCR		Access Rights 0. This bit defines the access rights for the HTU for accesses outside the region defined by the MP0S and MP0E registers for a single memory protection region configuration.
		0	HTU read access is allowed but write access will be signaled.
		1	Any access performed by the HTU is forbidden and will be signaled.

Table 18-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions (continued)

Bit	Field	Value	Description
0	REG0ENA		Region Enable 0
		0	The protection outside the memory region defined by the MP0S and MP0E registers is not enabled. This means the HTU can access any implemented memory space.
		1	The protection outside the memory region defined by the MP0S and MP0E registers is enabled. This means the HTU can perform any access within the region, but if it attempts to perform a forbidden access outside the region (according to the ACCR configuration), the access is signaled by the MPEFT0 flag, the number of the CP, which has caused the memory protection error, is captured to CPNUM0 if it is not currently frozen and an error is generated if it is enabled.

18.4.27 Memory Protection Start Address Register 0 (HTU MP0S)

This register configures the start address of memory protection region 0

Figure 18-40. Memory Protection Start Address Register 0 (HTU MP0S) [offset = 74h]

31	16
STARTADDRESS0	
R/WP-0	

15	2	1	0
STARTADDRESS0		0	0
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

Table 18-40. Memory Protection 0 Start Address Register (HTU MP0S) Field Descriptions

Bit	Field	Description
31-0	ISTARTADDRESS0	The start address defines at which main memory address the region begins. A memory protection error will be triggered, if the HTU accesses an address smaller than STARTADDRESS0 and the MPCS register is configured accordingly. The address is 32-bit aligned, so the 2 LSBs are not significant and will always read 0.

18.4.28 Memory Protection End Address Register (HTU MP0E)

Figure 18-41. Memory Protection End Address Register (HTU MP0E) [offset = 78h]

31	16
ENDADDRESS0	
R/WP-0	

15	2	1	0
ENDADDRESS0		0	0
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset

Table 18-41. Memory Protection End Address Register (HTU MP0E) Field Descriptions

Bit	Field	Description
31-0	ENDADDRESS0	The end address defines at which address the region ends. A memory protection error will be triggered, if the HTU accesses an address bigger than ENDADDRESS0 and the register bit MPCS register is configured accordingly. The address is 32-bit aligned, so the 2 LSBs are not significant and will always read 0. The effective end address is rounded up to the nearest word end address, that is, 0x200 = 0x203.

18.5 Double Control Packet Configuration Memory

All bits marked "reserved' are implemented in RAM and will be initialized to unknown values after power on. Reserved locations can be written and read, but should be written with 0 to ensure future compatibility. The HTU RAM can be cleared with the system RAM initialization function.

Table 18-42 provides a summary of the memory configuration. The base address for DCPs is FF4E 0000h.

Table 18-42. Double Control Packet Memory Map

Offset	Acronym	Register Description	Section
00h	HTU IFADDRA	Initial Full Address A Register	Section 18.5.1
04h	HTU IFADDRB	Initial Full Address B Register	Section 18.5.2
08h	HTU IHADDRCT	Initial N2HET Address and Control Register	Section 18.5.3
0Ch	HTU ITCOUNT	Initial Transfer Count Register	Section 18.5.4
100h	HTU CFADDRA	Current Full Address A Register	Section 18.5.5
104h	HTU CFADDRB	Current Full Address B Register	Section 18.5.6
108h	HTU CFCOUNT	Current Frame Count Register	Section 18.5.7

18.5.1 Initial Full Address A Register (HTU IFADDRA)

Figure 18-42. Initial Full Address A Register (HTU IFADDRA) [offset = 00h]

31	16
IFADDRA	
R/WP-X	

15	0
IFADDRA	
R/WP-X	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; X = value is unknown; -*n* = value after reset

Table 18-43. Initial Full Address A Register (HTU IFADDRA) Field Descriptions

Bit	Field	Description
31-0	IFADDRA	Initial Address of Buffer A in main memory. Initial (byte) address of buffer A placed in the main memory address range. Bits 0 and 1 are ignored by the logic, due to 32-bit alignment.

18.5.2 Initial Full Address B Register (HTU IFADDRB)

Figure 18-43. Initial Full Address B Register (HTU IFADDRB) [offset = 04h]

31	16
IFADDRB	
R/WP-X	

15	0
IFADDRB	
R/WP-X	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; X = value is unknown; -*n* = value after reset

Table 18-44. Initial Full Address B Register (HTU IFADDRB) Field Descriptions

Bit	Field	Description
31-0	IFADDRB	Initial Address of Buffer B in main memory. Initial (byte) address of buffer B placed in the main memory address range. Bits 0 and 1 are ignored by the logic, due to 32-bit alignment.

18.5.3 Initial N2HET Address and Control Register (HTU IHADDRCT)

Figure 18-44. Initial NHET Address and Control Register (HTU IHADDRCT) [offset = 08h]

31	24
Reserved	
R-0	

23	22	21	20	19	18	17	16
DIR	SIZE	ADDMH	ADDMF	TMBA		TMBB	
R/WP-X	R/WP-X	R/WP-X	R/WP-X	R/WP-X		R/WP-X	

15	13	12	2	1	0
Reserved		IHADDR		Reserved	
R-0		R/WP-X		R-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; X = value is unknown; -*n* = value after reset

Table 18-45. Initial N2HET Address and Control Register (HTU IHADDRCT) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23	DIR		Direction of Transfer
		0	N2HET address is read and main memory address is written.
		1	Main memory address is read and N2HET address is written.
22	SIZE		Size of Transferred Data
		0	32-bit transfer
		1	64-bit transfer
			64-bit transfer examples: If the N2HET address points to the N2HET instruction Control Field (CF), then the CF and Data Field (DF) will be transferred. If the N2HET address points to the Program Field (PF), then the PF and CF will be transferred.
21	ADDMH		Addressing Mode N2HET Address. This bit determines the N2HET address index from one to the next element of a frame.
		0	Increment by 16 bytes.
			Examples:
			If the initial N2HET address points to data field of instruction (n). Then the N2HET fields to be transferred by the elements of a frame are: data field of instruction (n), data field of instruction (n+1), data field of instruction (n+2) and so on. If the initial N2HET address points to control field of instruction (n), then the N2HET fields to be transferred by the elements of a frame are: control field of instruction (n), control field of instruction (n+1), control field of instruction (n+2), and so on.
		1	Increment by 8 bytes.
			This mode is intended to be used together with the 64-bit transfer size to load short N2HET instruction blocks into the N2HET RAM. So the sequence of transferred 64-bit elements could be: [PF and CF of instruction (n)], [DF and RF of instruction (n)], [PF and CF of instruction (n+1)], [DF and RF of instruction (n+1)], and so on.
20	ADDFM		Addressing Mode Main Memory Address
		0	Post-increment
			Note: When post-increment is selected the HTU will automatically increment by 4 bytes for a 32-bit data size and by 8 bytes for a 64-bit data size.
		1	Constant
19-18	TMBA		Transfer Mode for Buffer A
		0	One-Shot buffer mode
		1h	Circular buffer mode
		2h-3h	Auto-Switch mode
17-16	TMBB		Transfer Mode for Buffer B
		0	One-Shot buffer mode
		1h	Circular buffer mode
		2h-3h	Auto-Switch mode

Table 18-45. Initial N2HET Address and Control Register (HTU IHADDRCT) Field Descriptions (continued)

Bit	Field	Value	Description
15-13	Reserved	0	Reads return 0. Writes have no effect.
12-2	IHADDR		Initial N2HET Address The initial N2HET Address points to the N2HET field, which is the first element of the frame.The N2HET address (Bits 12:2) increments by 1 for each 32-bit N2HET field and starts with 0 at the first 32-bit field in the N2HET RAM. **Note:** When the HTU addresses the N2HET RAM it uses only the number of address bits required for the actual N2HET RAM size. If the N2HET address exceeds the actual N2HET RAM size, the unused MSB bits of the address will be ignored and the address rolls over to the start of the N2HET RAM.
1-0	Reserved	0	Reads return 0. Writes have no effect.

18.5.4 Initial Transfer Count Register (HTU ITCOUNT)

Figure 18-45. Initial Transfer Count Register (HTU ITCOUNT) [offset = 0Ch]

31	21	20	16
Reserved		IETCOUNT	
R-0		R/WP-X	

15	8	7	0
Reserved		IFTCOUNT	
R-0		R/WP-X	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; X = value is unknown; -*n* = value after reset

Table 18-46. Initial Transfer Count Register (HTU ITCOUNT) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reads return 0. Writes have no effect.
20-16	IETCOUNT		Initial Element Transfer Count Defines the number of element transfers.
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	IFTCOUNT		Initial Frame Transfer Count Defines the number of frame transfers.

18.5.5 Current Full Address A Register (HTU CFADDRA)

Figure 18-46. Current Full Address A Register (HTU CFADDRA) [offset = 100h]

31	16
CFADDRA	
R/WP-X	

15	0
CFADDRA	
R/WP-X	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; X = value is unknown; -*n* = value after reset

Table 18-47. Current Full Address A Register (HTU CFADDRA) Field Descriptions

Bit	Field	Description
31-0	CFADDRA	Current (byte) Address of Buffer A The current main memory address register is updated at the end of each frame. Therefore it points to the start address of the frame, which is the next to transfer, if currently no frame is transferred on this DCP. For an ongoing frame transfer, it points to the start address of this frame. After the last element of a buffer was transferred it will point to the buffer end address plus 0x4. The main purpose of the current full address registers for buffer A and buffer B (see next section) is to enable the software to find out the recently transferred element in the frozen buffer while the address of the active buffer increments. **Note:** A frame can be automatically stopped if any of the events listed in Conditions for Frame Transfer Interruption happens. If a frame is stopped before it could complete, then the current full address register is not updated and it will point to the start of the bad frame after the DCP was automatically disabled.

To transfer the first frame of buffer x, the information in the corresponding initial DCP RAM (IFADDRx, IHADDRCT, ITCOUNT) is loaded to an internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx).

This is valid for all of the following modes:

- Buffer x has reached it's end in circular mode and rolls back to its start address.
- CP x is enabled by a CPENA access (and corresponding BIM bit is 0).
- A CPENA access or auto-switch mode causes a switch from CP y to CP x

This means after starting the transfer to/from buffer x, CFADDRx and CFTCTx is not updated before the end of the first frame. So before the software switches from CP y to CP x using a write access to the CPENA register, it needs to initialize CFADDRx, CFTCTx. This allows the software to find out if the next request on CP x after the switching to CP x was delayed or never occurring.

18.5.6 Current Full Address B Register (HTU CFADDRB)

Figure 18-47. Current Full Address B Register (HTU CFADDRB) [offset = 104h]

31	16
CFADDRB	
R/WP-X	

15	0
CFADDRB	
R/WP-X	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; X = value is unknown; -*n* = value after reset

Table 18-48. Current Full Address B Register (HTU CFADDRB) Field Descriptions

Bit	Field	Description
31-0	CFADDRB	Current (byte) Address of Buffer B The current main memory address register is updated at the end of each frame. Therefore it points to the start address of the frame, which is the next to transfer, if currently no frame is transferred on this DCP. If currently a frame is transferred, then it points to the start address of this frame. After the last element of a buffer was transferred it will point to the buffer end address plus 0x4. The main purpose of the current full address registers for buffer A and buffer B (see next section) is to enable the software to find out the recently transferred element in the frozen buffer while the address of the active buffer increments. **Note:** A frame can be automatically stopped if any of the events listed in Conditions for Frame Transfer Interruption happens. If a frame is stopped before it could complete, then the current full address register is not updated and it will point to the start of the bad frame after the DCP was automatically disabled.

To transfer the first frame of buffer x, the information in the corresponding initial DCP RAM (IFADDRx, IHADDRCT, ITCOUNT) is loaded to an internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx).

This is valid for all of the following modes:

- Buffer x has reached it's end in circular mode and rolls back to its start address.
- CP x is enabled by a CPENA access (and corresponding BIM bit is 0).
- A CPENA access or auto-switch mode causes a switch from CP y to CP x

This means after starting the transfer to/from buffer x, CFADDRx and CFTCTx is not updated before the end of the first frame. So before the software switches from CP y to CP x using a write access to the CPENA register, it needs to initialize CFADDRx, CFTCTx. This allows the software to find out if the next request on CP x after the switching to CP x was delayed or never occurring.

18.5.7 Current Frame Count Register (HTU CFCOUNT)

The current frame count register enables the software to find out the recent frame in the buffer while the counter of the active buffer decrements.

Figure 18-48. Current Frame Count Register (HTU CFCOUNT) [offset = 108h]

31	24	23	16
Reserved		CFTCTA	
R-0		R/WP-X	

15	8	7	0
Reserved		CFTCTB	
R-0		R/WP-X	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; X = value is unknown; -*n* = value after reset

Table 18-49. Current Frame Count Register (HTU CFCOUNT) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	CFTCTA		Current Frame Transfer Count for CP A. It is updated at the end of each frame.
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	CFTCTB		Current Frame Transfer Count for CP B. It is updated at the end of each frame.

18.6 Examples

18.6.1 Application Examples for Setting the Transfer Modes of CP A and B of a DCP

Table 18-50. Application Examples for Setting the Transfer Modes of CP A and B of a DCP

CP A	CP B	
One shot	Not used	Buffer A can be used as a one-shot buffer. A buffer full interrupt enabled for CP A can signal reaching the end of the buffer.
Auto switch	One shot	Can double the buffer size for a one-shot buffer. A buffer full interrupt enabled for CP B can signal reaching the end of the buffer.
Circular	Circular	The CPU can switch the buffers at arbitrary times. It will fill or read the frozen buffer during the other buffer is filled or read by the HTU. Interrupts are not required for this case.
Auto switch	Auto switch	Buffer full interrupts (enabled for CP A and B) signal when the end of a buffer is reached. After one buffer is completed the according CPU interrupt routine will read or refill this buffer. At the same time the other buffer is read or filled by the HTU. Here the time when the buffer must be read is determined by the time of the interrupt (determined by the frequency of the N2HET transfer requests).

18.6.2 Software Example Sequence Assuming Circular Mode for Both CP A and B

The example assumes the N2HET address to be read and the main memory address to be written.

I1 CPU initializes initial DCP: IFADDRA, IFADDRB, IHADDRCT, ITCOUNT
I2 CPU clears current DCP: CFADDRA, CFADDRB, CFTCTA, CFTCTB
I3 CPU clears BFINTFL flag of CP A and B
I4 Enable CP A with the CPENA register. Now the HTU fills buffer A

After some time the CPU intends to read buffer A:

A1 CPU enables CP B and disables CP A by writing to the CPENA register. After this switch, the HTU fills buffer B. Filling buffer B starts with its initial full address and initial frame counter.
A2 CPU waits for CP A busy bit equals 0
A3 Optional: CPU verifies that the CP A request lost flag is not set. The bus error flag of CP A could also be checked.
A4 CPU reads the frozen CFTCTA, which indicates the fill level in the buffer
A5 CPU sets current CP A (CFTCTA and/or CFADDRA) to 0. This allows to find out if any request has happened during the next time buffer A is active.
A6 CPU reads BFINTFL flag of buffer A
A7 CPU clears the BFINTFL flag of buffer A. This is an initialization for the next time buffer A is used.
A8 CPU reads valid values of frozen buffer A. After reading the CPU does not need to clear the frozen buffer A.

After some time the CPU intends to read buffer B:

B1	CPU enables CP A and disables CP B by writing to the CPENA register. After this switch, the HTU fills buffer A. Filling buffer A starts with its initial full address and initial frame counter.
B2	CPU waits for CP B busy bit equals 0
B3	Optional: CPU verifies that the CP B request lost flag is not set. The bus error flag of CP B could also be checked.
B4	CPU reads the frozen CFTCTB, which indicates the fill level in the buffer
B5	CPU sets current CP B (CFTCTB and/or CFADDRB) to 0. This allows to find out if any request has happened during the next time buffer B is active.
B6	CPU reads BFINTFL flag of buffer B
B7	CPU clears the BFINTFL flag of buffer B. This is an initialization for the next time buffer B is used.
B8	CPU reads valid values of frozen buffer B. After reading the CPU does not need to clear the frozen buffer B.

After some time the CPU intends to read buffer A:

A1) ... see above...

NOTE: The buffer full interrupt doesn't need to be enabled. The BFINTFL flag is used to indicate a circular overrun of the buffer. If the BFINTFL flag is set, also the buffer section after the frozen full address could be read.

Steps A3 and B3 in the example sequence above imply that request lost interrupts are disabled. The example below assumes that request lost interrupts are enabled.

Request lost detection with interrupt enabled.

18.6.3 Example of an Interrupt Dispatch Flow for a Request Lost Interrupt

- A request lost occurs and the interrupt routine starts.
- Reading INTOFFx.INTYPEx shows that RLOSTFL is the interrupt source.
- Reading INTOFFx.CPOFFx = Ah shows that DCP 5 / CP A has caused the RLOSTFL interrupt. The hardware automatically clears bit (2·5+0) in RLOSTFL.
- Reading RLOSTFL= 84h shows that also another request lost event happened on DCP 1 / CP A [bit (2·1+0)] and on DCP 3 / CP B [bit (2·3+1)] at the same time or after the request lost occurred on DCP 5 / CP A.
- Writing back 84h to RLOSTFL clears bits 2 and 7 and the according pending interrupts.

General-Purpose Input/Output (GIO) Module

This chapter describes the general-purpose input/output (GIO) module. The GIO module provides the family of devices with input/output (I/O) capability. The I/O pins are bidirectional and bit-programmable. The GIO module also supports external interrupt capability.

Topic **Page**

19.1 Overview

The GIO module offers general-purpose input and output capability. It supports up to eight 8-bit ports for a total of up to 64 GIO terminals. Each of these 64 terminals can be independently configured as input or output and configured as required by the application. The GIO module also supports generation of interrupts whenever a rising edge or falling edge or any toggle is detected on up to 32 of these GIO terminals. Refer to the device datasheet for identifying the number of GIO ports supported and the GIO terminals capable of generating an interrupt.

The main features of the GIO module are summarized as follows:

- Allows each GIO terminal to be configured for general-purpose input or output functions
- Supports programmable pull directions on each input GIO terminal
- Supports GIO output in push/pull or open-drain modes
- Allows up to 32 GIO terminals to be used for generating interrupt requests

19.2 Quick Start Guide

The GIO module comprises two separate components: an input/output (I/O) block and an interrupt generation block. Figure 19-1 and Figure 19-2 show what you should do after reset to configure the GIO module as I/O or for generating interrupts.

In GIO interrupt service routine, you shall read the GIO offset register (GIOOFF1 or GIOOFF2, depending on high-/low-level interrupt) to clear the flag and find the pending interrupt GIO channel.

Figure 19-1. I/O Function Quick Start Flow Chart

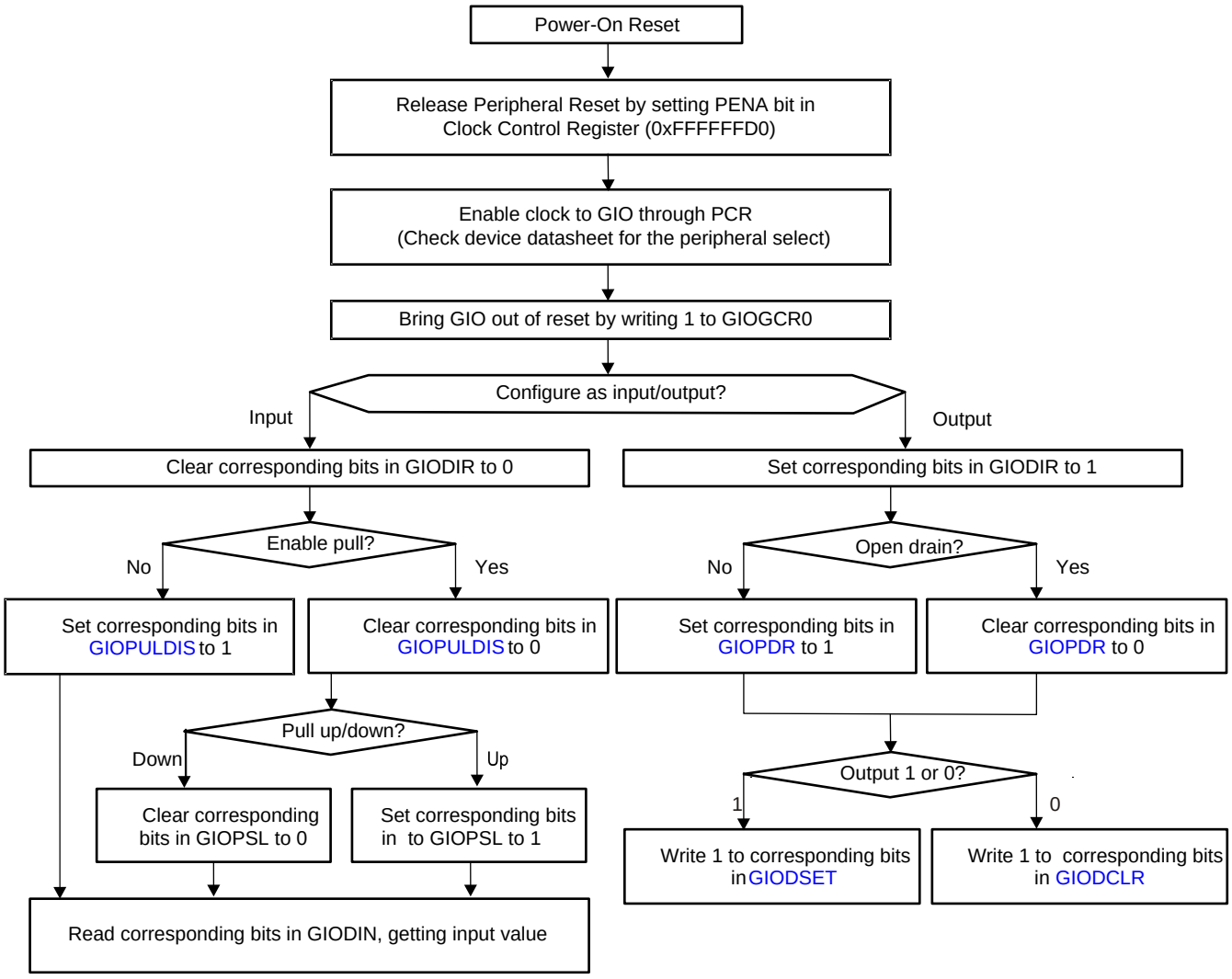
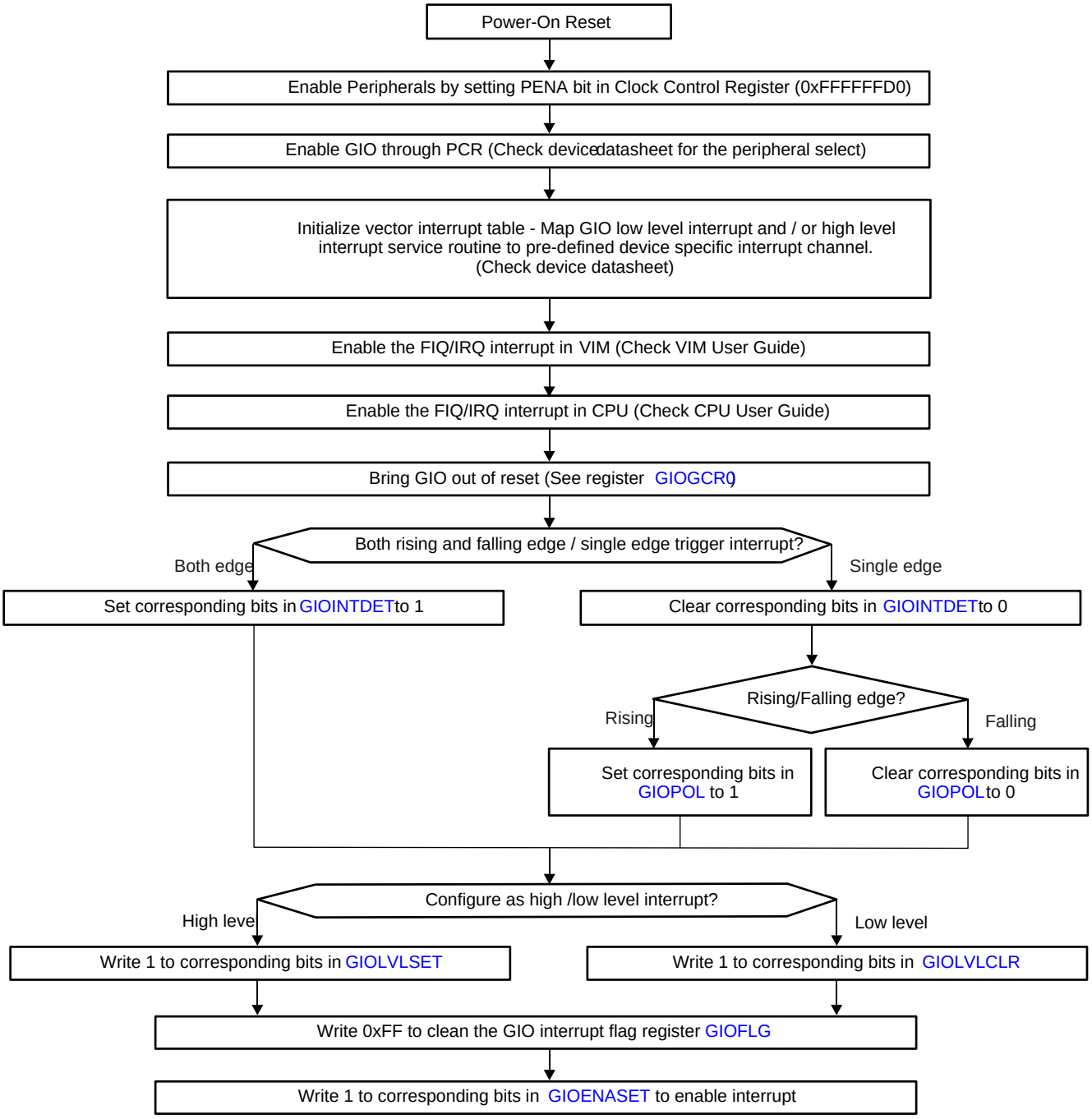


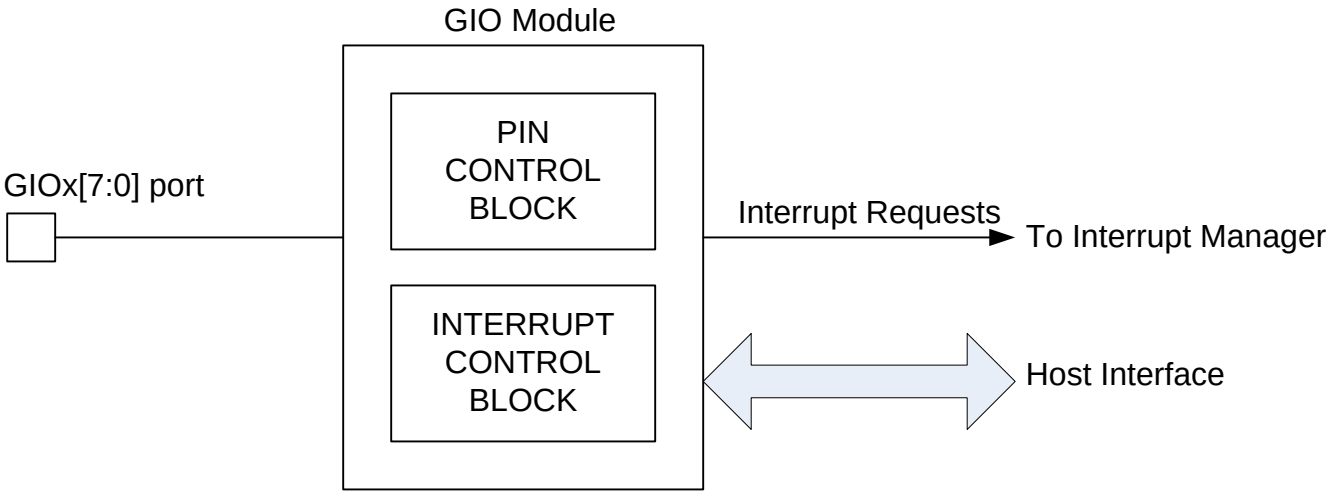
Figure 19-2. Interrupt Generation Function Quick Start Flow Chart



19.3 Functional Description of GIO Module

As shown in Figure 19-3, the GIO module comprises of two separate components: an input/output (I/O) block and an interrupt block.

Figure 19-3. GIO Module Diagram



19.3.1 I/O Functions

The I/O block allows each GIO terminal to be configured for use as a general-purpose input or output in the application. The GIO module supports multiple registers to control the various aspects of the input and output functions. These are described as follows.

- Data direction (GIODIR)

  Configures GIO terminal(s) as input (default) or output through the GIODIRx registers.

- Data input (GIODIN)

  Reflects the logic level on GIO terminals in the GIODINx registers. A high voltage (V_{IH} or greater) applied to the pin causes a high value (1) in the data input register (GIODIN[7:0]). When a low voltage (V_{IL} or less) is applied to the pin, the data input register reads a low value (0). The V_{IH} and V_{IL} values are device specific and can be found in the device datasheet.

- Data output (GIODOUT)

  Configures the logic level to be output on GIO terminal(s) configured as outputs. A low value (0) written to the data output register forces the pin to a low output voltage (V_{OL} or lower). A high value (1) written to the data output register (GIODOUTx) forces the pin to a high output voltage (V_{OH} or higher) if the open drain functionality is disabled (GIOPDRx[7:0]). If open drain functionality is enabled, a high value (1) written to the data output register forces the pin to a high-impedance state (Z).

- Data set (GIODSET)

  Allows logic HIGH to be output on GIO terminal(s) configured as outputs by writing 1's to the required bits in the GIODSETx registers. If open drain functionality is enabled, a high value (1) written to the data output register forces the pin to a high-impedance state (Z). The GIODSETx registers eliminate the need for the application to perform a read-modify-write operation when it needs to set one or more GIO pin(s).

- Data clear (GIODCLR)

  Allows logic LOW to be output on GIO terminal(s) configured as outputs by writing 1s to the required bits in the GIODCLRx registers. The GIODCLRx registers eliminate the need for the application to perform a read-modify-write operation when it needs to clear one or more GIO pin(s).

- Open drain (GIOPDR)

  Open drain functionality is enabled or disabled (default) using the open drain register GIOPDR[7:0] register. If open-drain mode output is enabled on a pin, a high value (1) written to the data output register (GIODOUTx[7:0]) forces the pin to a high impedance state (Z).

- Pull disable (GIOPULDIS)

  Disables the internal pull on GIO terminal(s) configured as inputs by writing to the GIOPULDISx registers.
- Pull select (GIOPSL)

  Selects internal pull down (default) or pull up on GIO terminal(s) configured as inputs by writing to the GIOPULSELx registers.

Refer to the specific device's datasheet to identify the number of GIO ports as well as the input and output functions supported. Some devices may not support the programmable pull controls. In that case, the pull disable and the pull select register controls will not work.

19.3.2 Interrupt Function

The GIO module supports up to 32 terminals to be configured for generating an interrupt to the host processor through the Vectored Interrupt Manager (VIM). The main functions of the interrupt block are:

- Select the GIO pin(s) that is/are used to generate interrupt(s)

  This is done via the interrupt enable set and clear registers, GIOENASET and GIOENACLR.
- Select the edge on the selected GIO pin(s) that is/are used to generate interrupt(s): rising/falling/both

  Rising or falling edge can be selected via the GIOPOL register. If interrupt is required to be generated on both rising and falling edges, this can be configured via the GIOINTDET register.
- Select the interrupt priority

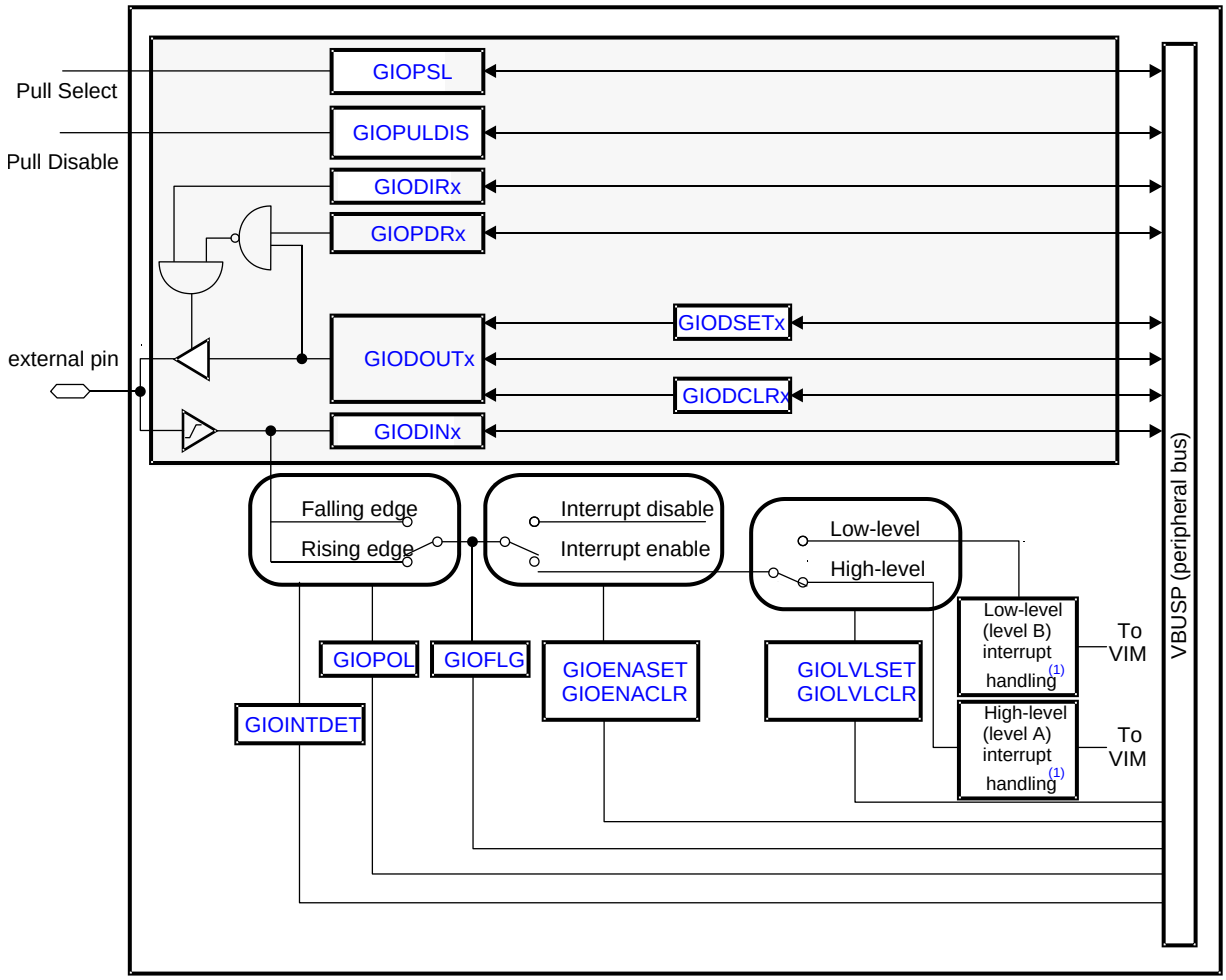
  Low- or high-level interrupt can be selected through the GIOLVLSET and GIOLVLCLR registers.
- Individual interrupt flags are set in the GIOFLG register

The terminals on GIO ports A through D are all interrupt-capable and can be used to handle either general I/O functions or interrupt requests. Each interrupt request can be connected to the VIM at one of two different levels – High (or A) and Low (or B), depending on the VIM channel number. The VIM has an inherent priority scheme so that a request on a lower number channel has a higher priority than a request on a higher number channel. Refer the device datasheet to identify the VIM channel numbers for the GIO level A and level B interrupt requests. Also note that the interrupt priority of level A and level B interrupt handling blocks can be re-programmed in the VIM.

19.3.3 GIO Block Diagram

The GIO block diagram (Figure 19-4) represents the flow of information through a pin. The shaded area corresponds to the I/O block; the unshaded area corresponds to the interrupt block.

Figure 19-4. GIO Block Diagram



(1) A single low-level-interrupt-handling block and a single high-level-interrupt-handling block service all of the interrupt-capable external pins, but only one pin can be serviced by an interrupt block at a time.

19.4 Device Modes of Operation

The GIO module behaves differently in different modes of operation. There are two main modes:

- Emulation mode
- Power-down mode (low-power mode)

19.4.1 Emulation Mode

Emulation mode is used by debugger tools to stop the CPU at breakpoints to read registers.

NOTE: Emulation Mode and Emulation Registers

Emulation mode is a mode of operation of the device and is separate from the GIO emulation registers (GIOEMU1 and GIOEMU2). The contents of these emulation registers are identical to the contents of GIO offset registers (GIOOFF1 and GIOOFF2). Both emulation registers and GIO offset registers are NOT cleared when they are read in emulation mode. GIO offset registers are cleared when they are read in normal mode (other than emulation mode). The emulation registers are NOT cleared when they are read in normal mode. The intention for the emulation registers is that software can use them without clearing the flags.

During emulation mode:

- External interrupts are not captured because the VIM is unable to service interrupts.
- Any register can be read without affecting the state of the system.
- A write to a register still does affect the state of the system.

19.4.2 Power-Down Mode (Low-Power Mode)

In power-down mode, the clock signal to the GIO module is disabled. Thus, there is no switching and the only current draw comes from leakage current. In power-down mode, interrupt pins become level-sensitive rather than edge-sensitive. The polarity bit changes function from falling-edge-triggered to low-level-triggered and rising-edge-triggered to high-level-triggered. A corresponding level on an interrupt pin pulls the module out of low-power mode, if the interrupt is also enabled to wake up the device out of a low-power mode.

19.4.2.1 Module-Level Power Down

The GIO module can be placed into a power down state by disabling the GIO peripheral module via the appropriate bit in the peripheral power down register. Please refer to the Peripheral Central Resource Registers (Section 2.5.3) for details.

19.4.2.2 Device-Level Power Down

The entire device can be placed in one of the pre-defined low-power modes: doze, snooze, or sleep using the clock source and clock domain disable registers in the system module.

19.5 GIO Control Registers

Table 19-1 shows the summary of the GIO registers. The registers are accessible in 8-, 16-, and 32-bit reads or writes.

The start address for the GIO module is FFF7 BC00h.

The GIO module supports up to 8 ports. Refer to your device-specific data manual to identify the actual number of GIO ports and the number of pins in each GIO port implemented on this device.

The GIO module supports up to 4 interrupt-capable ports. Refer to the device datasheet to identify the actual number of interrupt-capable GIO ports and the number of pins in each GIO port implemented on this device.

Table 19-1. GIO Control Registers

Offset	Acronym	Register Description	Section
00h	GIOGCR0	GIO Global Control Register	Section 19.5.1
08h	GIOINTDET	GIO Interrupt Detect Register	Section 19.5.2
0Ch	GIOPOL	GIO Interrupt Polarity Register	Section 19.5.3
10h	GIOENASET	GIO Interrupt Enable Set Register	Section 19.5.4.1
14h	GIOENACLR	GIO Interrupt Enable Clear Register	Section 19.5.4.2
18h	GIOLVLSET	GIO Interrupt Priority Set Register	Section 19.5.5.1
1Ch	GIOLVLCLR	GIO Interrupt Priority Clear Register	Section 19.5.5.2
20h	GIOFLG	GIO Interrupt Flag Register	Section 19.5.6
24h	GIOOFF1	GIO Offset 1 Register	Section 19.5.7
28h	GIOOFF2	GIO Offset 2 Register	Section 19.5.8
2Ch	GIOEMU1	GIO Emulation 1 Register	Section 19.5.9
30h	GIOEMU2	GIO Emulation 2 Register	Section 19.5.10
34h	GIODIRA	GIO Data Direction Register	Section 19.5.11
38h	GIODINA	GIO Data Input Register	Section 19.5.12
3Ch	GIODOUTA	GIO Data Output Register	Section 19.5.13
40h	GIODSETA	GIO Data Set Register	Section 19.5.14
44h	GIODCLRA	GIO Data Clear Register	Section 19.5.15
48h	GIOPDRA	GIO Open Drain Register	Section 19.5.16
4Ch	GIOPULDISA	GIO Pull Disable Register	Section 19.5.17
50h	GIOPSLA	GIO Pull Select Register	Section 19.5.18
54h	GIODIRB	GIO Data Direction Register	Section 19.5.11
58h	GIODINB	GIO Data Input Register	Section 19.5.12
5Ch	GIODOUTB	GIO Data Output Register	Section 19.5.13
60h	GIODSETB	GIO Data Set Register	Section 19.5.14
64h	GIODCLRB	GIO Data Clear Register	Section 19.5.15
68h	GIOPDRB	GIO Open Drain Register	Section 19.5.16
6Ch	GIOPULDISB	GIO Pull Disable Register	Section 19.5.17
70h	GIOPSLB	GIO Pull Select Register	Section 19.5.18

19.5.1 GIO Global Control Register (GIOGCR0)

The GIOGCR0 register contains one bit that controls the module reset status. Writing a 0 to this bit puts the module in a reset state. After system reset, this bit must be set to 1 before configuring any other register of the GIO module. Figure 19-5 and Table 19-2 describe this register.

Figure 19-5. GIO Global Control Register (GIOGCR0) [offset = 00h]

31	16
Reserved	
R-0	

15	1	0
Reserved		RESET
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 19-2. GIO Global Control Register (GIOGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RESET		GIO reset.
		0	The GIO is in reset state.
		1	The GIO is operating normally.

NOTE: Note that putting the GIO module in reset state is not the same as putting it in a low-power state.

19.5.2 GIO Interrupt Detect Register (GIOINTDET)

The GIO module supports generation of an interrupt request to CPU when a rising edge, falling edge, or both edges is detected on one or more GIO pin(s). The GIOINTDET register allows both rising and falling edges to be detected, while the GIOPOL register allows the application to define whether a rising edge or a falling edge is to be detected. Figure 19-6 and Table 19-3 describe this register.

Figure 19-6. GIO Interrupt Detect Register (GIOINTDET) [offset = 08h]

31	24	23	16
GIOINTDET 3		GIOINTDET 2	
R/W-0		R/W-0	

15	8	7	0
GIOINTDET 1		GIOINTDET 0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 19-3. GIO Interrupt Detect Register (GIOINTDET) Field Descriptions

Bit	Field	Value	Description
31-24	GIOINTDET 3		Interrupt detection select for pins GIOD[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register (GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.
23-16	GIOINTDET 2		Interrupt detection select for pins GIOC[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register (GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.
15-8	GIOINTDET 1		Interrupt detection select for pins GIOB[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register (GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.
7-0	GIOINTDET 0		Interrupt detection select for pins GIOA[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register (GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.

19.5.3 GIO Interrupt Polarity Register (GIOPOL)

The GIOPOL register configures the polarity of the edge, rising edge or falling edge, that needs to be detected. When the device is in low-power mode, the GIOPOL register controls the ***level***, high or low, which will be detected by the GIO module. Figure 19-7 and Table 19-4 describe this register.

Figure 19-7. GIO Interrupt Polarity Register (GIOPOL) [offset = 0Ch]

31	24	23	16
GIOPOL 3		GIOPOL 2	
R/W-0		R/W-0	

15	8	7	0
GIOPOL 1		GIOPOL 0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 19-4. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions

Bit	Field	Value	Description
31-24	GIOPOL 3		Interrupt polarity select for pins GIOD[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
23-16	GIOPOL 2		Interrupt polarity select for pins GIOC[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
15-8	GIOPOL 1		Interrupt polarity select for pins GIOB[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
7-0	GIOPOL 0		Interrupt polarity select for pins GIOA[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.

19.5.4 GIO Interrupt Enable Registers (GIOENASET and GIOENACLR)

The GIOENASET and GIOENACLR registers control which interrupt-capable pins are actually configured as interrupts. If the interrupt is enabled, the rising edge, falling edge, or both edges on the selected pin lead to an interrupt request.

19.5.4.1 GIOENASET Register

Figure 19-8 and Table 19-5 describe this register.

NOTE: Enabling Interrupt at the Device Level

The interrupt channel in the Vectored Interrupt Manager (VIM) must be enabled for the interrupt request to be forwarded to the CPU. Additionally, the ARM CPU (CPSR bit 7 or 6) must be cleared to respond to interrupt requests (IRQ/FIQ).

Figure 19-8. GIO Interrupt Enable Set Register (GIOENASET) [offset = 10h]

31	24	23	16
GIOENASET 3		GIOENASET 2	
R/W-0		R/W-0	

15	8	7	0
GIOENASET 1		GIOENASET 0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 19-5. GIO Interrupt Enable Set Register (GIOENASET) Field Descriptions

Bit	Field	Value	Description
31-24	GIOENASET 3		Interrupt enable for pins GIOD[7:0]
		0	Read: The interrupt is disabled.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled.
			Write: Enables the interrupt.
23-16	GIOENASET 2		Interrupt enable for pins GIOC[7:0]
		0	Read: The interrupt is disabled.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled.
			Write: Enables the interrupt.
15-8	GIOENASET 1		Interrupt enable for pins GIOB[7:0]
		0	Read: The interrupt is disabled.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled.
			Write: Enables the interrupt.
7-0	GIOENASET 0		Interrupt enable for pins GIOA[7:0]
		0	Read: The interrupt is disabled.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled.
			Write: Enables the interrupt.

19.5.4.2 GIOENACLR Register

This register disables the interrupt. Figure 19-9 and Table 19-6 describe this register.

Figure 19-9. GIO Interrupt Enable Clear Register (GIOENACLR) [offset = 14h]

31	24	23	16
GIOENACLR 3		GIOENACLR 2	
R/W-0		R/W-0	

15	8	7	0
GIOENACLR 1		GIOENACLR 0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 19-6. GIO Interrupt Enable Clear Register (GIOENACLR) Field Descriptions

Bit	Field	Value	Description
31-24	GIOENACLR 3		Interrupt disable for pins GIOD[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
23-16	GIOENACLR 2		Interrupt disable for pins GIOC[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
15-8	GIOENACLR 1		Interrupt disable for pins GIOB[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
7-0	GIOENACLR 0		Interrupt disable for pins GIOA[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.

19.5.5 GIO Interrupt Priority Registers (GIOLVLSET and GIOLVLCLR)

The GIOLVLSET and GIOLVLCLR registers configure the interrupts as high-level (level A) or low-level (level B) going to the Vectored Interrupt Manager (VIM). Each interrupt is individually configured.

- The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.
- The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.

NOTE: The GIO module can generate two interrupt requests. These are connected to two separate channels on the Vectored Interrupt Manager (VIM). The lower-numbered VIM channels are higher priority. The GIO interrupt connected to a lower-number channel is the high-level (also called level A) GIO interrupt, while the GIO interrupt connected to a higher-number channel is the low-level (also called level B) GIO interrupt.

19.5.5.1 GIOLVLSET Register

The GIOLVLSET register is used to configure an interrupt as a high-level interrupt going to the VIM. An interrupt can be configured as a high-level interrupt by writing a 1 into the corresponding bit of the GIOLVLSET register. Writing a 0 has no effect. Figure 19-10 and Table 19-7 describe this register.

Figure 19-10. GIO Interrupt Priority Register (GIOLVLSET) [offset = 18h]

31	24	23	16
GIOLVLSET 3		GIOLVLSET 2	
R/W-0		R/W-0	

15	8	7	0
GIOLVLSET 1		GIOLVLSET 0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 19-7. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions

Bit	Field	Value	Description
31-24	GIOLVLSET 3		GIO high-priority interrupt for pins GIOD[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.
23-16	GIOLVLSET 2		GIO high-priority interrupt for pins GIOC[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.
15-8	GIOLVLSET 1		GIO high-priority interrupt for pins GIOB[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.

Table 19-7. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions (continued)

Bit	Field	Value	Description
7-0	GIOLVLSET 0		GIO high-priority interrupt for pins GIOA[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.

19.5.5.2 GIOLVLCLR Register

The GIOLVLCLR register is used to configure an interrupt as a low-level interrupt going to the VIM. An interrupt can be configured as a low-level interrupt by writing a 1 into the corresponding bit of the GIOLVLCLR register. Writing a 0 has no effect. Figure 19-11 and Table 19-8 describe this register.

Figure 19-11. GIO Interrupt Priority Register (GIOLVLCLR) [offset = 1Ch]

31		24	23		16
GIOLVLCLR 3			GIOLVLCLR 2		
R/W-0			R/W-0		
15		8	7		0
GIOLVLCLR 1			GIOLVLCLR 0		
R/W-0			R/W-0		

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 19-8. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions

Bit	Field	Value	Description
31-24	GIOLVLCLR 3		GIO low-priority interrupt for pins GIOD[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.
23-16	GIOLVLCLR 2		GIO low-priority interrupt for pins GIOC[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.
15-8	GIOLVLCLR 1		GIO low-priority interrupt for pins GIOB[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.
7-0	GIOLVLCLR 0		GIO low-priority interrupt for pins GIOA[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.

19.5.6 GIO Interrupt Flag Register (GIOFLG)

The GIOFLG register contains flags indicating that the transition edge (as set in GIOINTDET and GIOPOL registers) has occurred. The flag can be cleared by the CPU writing a 1 to the flag that is set. The flag is also cleared by reading the appropriate interrupt offset register (GIOOFF1 or GIOOFF2). Figure 19-12 and Table 19-9 describe this register.

Figure 19-12. GIO Interrupt Flag Register (GIOFLG) [offset = 20h]

31	24	23	16
GIOFLG 3		GIOFLG 2	
R/W1C-0		R/W1C-0	

15	8	7	0
GIOFLG 1		GIOFLG 0	
R/W1C-0		R/W1C-0	

LEGEND: R/W = Read/Write; W1C = Write 1 to clear; -*n* = value after reset

Table 19-9. GIO Interrupt Flag Register (GIOFLG) Field Descriptions

Bit	Field	Value	Description
31-24	GIOFLG 3		GIO flag for pins GIOD[7:0]
		0	Read: A transition has not occurred since the last clear.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred.
			Write: The corresponding bit is cleared to 0.
			Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.
23-16	GIOFLG 2		GIO flag for pins GIOC[7:0]
		0	Read: A transition has not occurred since the last clear.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred.
			Write: The corresponding bit is cleared to 0.
			Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.
15-8	GIOFLG 1		GIO flag for pins GIOB[7:0]
		0	Read: A transition has not occurred since the last clear.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred.
			Write: The corresponding bit is cleared to 0.
			Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.
7-0	GIOFLG 0		GIO flag for pins GIOA[7:0]
		0	Read: A transition has not occurred since the last clear.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred.
			Write: The corresponding bit is cleared to 0.
			Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.

NOTE: An interrupt flag gets set when the selected transition happens on the corresponding GIO pin ***regardless of whether the interrupt generation is enabled or not***. It is recommended to clear a flag before enabling the interrupt generation for a transition on the corresponding GIO pin.

19.5.7 GIO Offset Register 1 (GIOOFF1)

The GIOOFF1 register provides a numerical offset value that represents the pending external interrupt with high priority. The offset value can be used to locate the position of the interrupt routine in a vector table in application software. Figure 19-13 and Table 19-10 describe this register.

NOTE: Reading this register clears it, GIOEMU1 and the corresponding flag bit in the GIOFLG register. However, in emulation mode, a read to this register does not clear any register or flag. If more than one GIO interrupts are pending, then reading the GIOOFF1 register will change the contents of GIOOFF1 and GIOEMU1 to show the offset value for the next highest-priority pending interrupt. The application can choose to service all GIO interrupts from the same service routine by continuing to read the GIOOFF1 register until it reads zeros.

Figure 19-13. GIO Offset 1 Register (GIOOFF1) [offset = 24h]

31	16
Reserved	
R-0	

15	6	5	0
Reserved		GIOOFF1	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 19-10. GIO Offset 1 Register (GIOOFF1) Field Descriptions

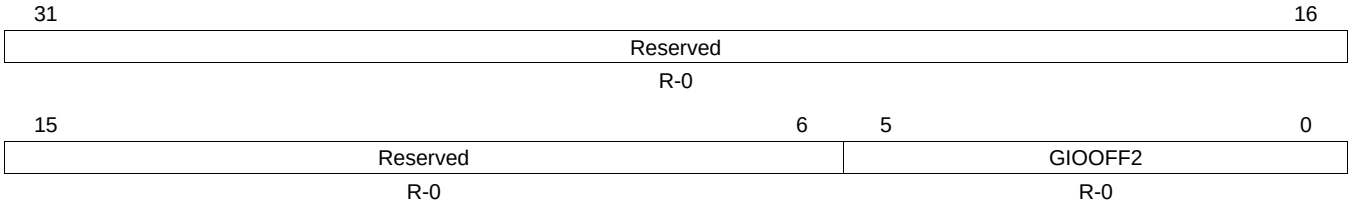
Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	GIOOFF1		GIO offset 1. These bits index the currently pending high-priority interrupt. This register and the flag bit (in the GIOFLG register) are also cleared when this register is read, except in emulation mode.
		0	No interrupt is pending.
		1h	Interrupt 0 (corresponding to GIOA0) is pending with a high priority.
		:	:
		8h	Interrupt 7 (corresponding to GIOA7) is pending with a high priority.
		9h	Interrupt 8 (corresponding to GIOB0) is pending with a high priority.
		:	:
		10h	Interrupt 16 (corresponding to GIOB7) is pending with a high priority.
		:	:
		20h	Interrupt 32 (corresponding to GIOD7) is pending with a high priority.
		21h-3Fh	Reserved

19.5.8 GIO Offset B Register (GIOOFF2)

The GIOOFF2 register provides a numerical offset value that represents the pending external interrupt with low priority. The offset value can be used to locate the position of the interrupt routine in a vector table in application software. Figure 19-14 and Table 19-11 describe this register.

NOTE: Reading this register clears it, GIOEMU2 and the corresponding flag bit in the GIOFLG register. However, in emulation mode, a read to this register does not clear any register or flag. If more than one GIO interrupts are pending, then reading the GIOOFF1 register will change the contents of GIOOFF2 and GIOEMU2 to show the offset value for the next highest-priority pending interrupt. The application can choose to service all GIO interrupts from the same service routine by continuing to read the GIOOFF1 register until it reads zeros.

Figure 19-14. GIO Offset 2 Register (GIOOFF2) [offset = 28h]



LEGEND: R = Read only; -*n* = value after reset

Table 19-11. GIO Offset 2 Register (GIOOFF2) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	GIOOFF2		GIO offset 2. These bits index the currently pending low-priority interrupt. This register and the flag bit (in the GIOFLG register) are also cleared when this register is read, except in emulation mode.
		0	No interrupt is pending.
		1h	Interrupt 0 (corresponding to GIOA0) is pending with a low priority.
		:	:
		8h	Interrupt 7 (corresponding to GIOA7) is pending with a low priority.
		9h	Interrupt 8 (corresponding to GIOB0) is pending with a low priority.
		:	:
		10h	Interrupt 16 (corresponding to GIOB7) is pending with a low priority.
		:	:
		20h	Interrupt 32 (corresponding to GIOD7) is pending with a low priority.
		21h-3Fh	Reserved

19.5.9 GIO Emulation A Register (GIOEMU1)

The GIOEMU1 register is a read-only register. The contents of this register are identical to the contents of GIOOFF1. The intention for the this register is that software can use it without clearing the flags. Figure 19-15 and Table 19-12 describe this register.

NOTE: The corresponding flag in the GIOFLG register is not cleared when the GIOEMU1 register is read.

Figure 19-15. GIO Emulation 1 Register (GIOEMU1) [offset = 2Ch]

31	16
Reserved	
R-0	

15	6	5	0
Reserved		GIOEMU1	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 19-12. GIO Emulation 1 Register (GIOEMU1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	GIOEMU1		GIO offset emulation 1. These bits index the currently pending high-priority interrupt. No register or flag is cleared by reading this register.
		0	No interrupt is pending.
		1h	Interrupt 0 (corresponding to GIOA0) is pending with a high priority.
		:	:
		8h	Interrupt 7 (corresponding to GIOA7) is pending with a high priority.
		9h	Interrupt 8 (corresponding to GIOB0) is pending with a high priority.
		:	:
		10h	Interrupt 16 (corresponding to GIOB7) is pending with a high priority.
		:	:
		20h	Interrupt 32 (corresponding to GIOD7) is pending with a high priority.
		21h-3Fh	Reserved

19.5.10 GIO Emulation B Register (GIOEMU2)

The GIOEMU2 register is a read-only register. The contents of this register are identical to the contents of GIOOFF2. The intention for the this register is that software can use it without clearing the flags. Figure 19-16 and Table 19-13 describe this register.

NOTE: The corresponding flag in the GIOFLG register is not cleared when the GIOEMU2 register is read.

Figure 19-16. GIO Emulation 2 Register (GIOEMU2) [offset = 30h]

31	16
Reserved	
R-0	

15	6	5	0
Reserved		GIOEMU2	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

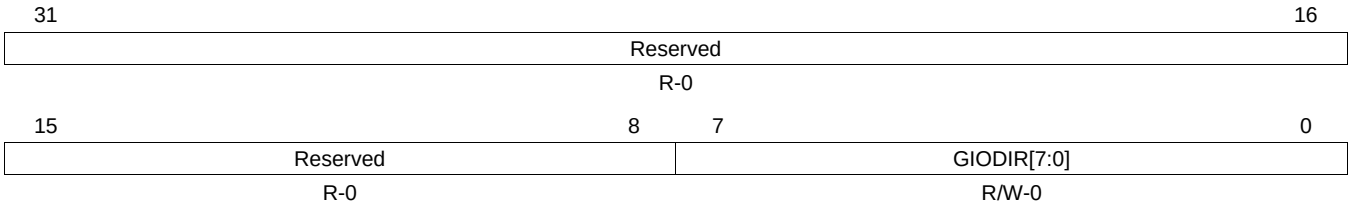
Table 19-13. GIO Emulation 2 Register (GIOEMU2) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	GIOEMU2		GIO offset emulation 2. These bits index the currently pending low-priority interrupt. No register or flag is cleared by reading this register.
		0	No interrupt is pending.
		1h	Interrupt 0 (corresponding to GIOA0) is pending with a low priority.
		:	:
		8h	Interrupt 7 (corresponding to GIOA7) is pending with a low priority.
		9h	Interrupt 8 (corresponding to GIOB0) is pending with a low priority.
		:	:
		10h	Interrupt 16 (corresponding to GIOB7) is pending with a low priority.
		:	:
		20h	Interrupt 32 (corresponding to GIOD7) is pending with a low priority.
		21h-3Fh	Reserved

19.5.11 GIO Data Direction Registers (GIODIR[A-B])

The GIODIR register controls whether the pins of a given port are configured as inputs or outputs. Figure 19-17 and Table 19-14 describe this register.

Figure 19-17. GIO Data Direction Registers (GIODIR[A-B]) [offset = 34h, 54h]



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

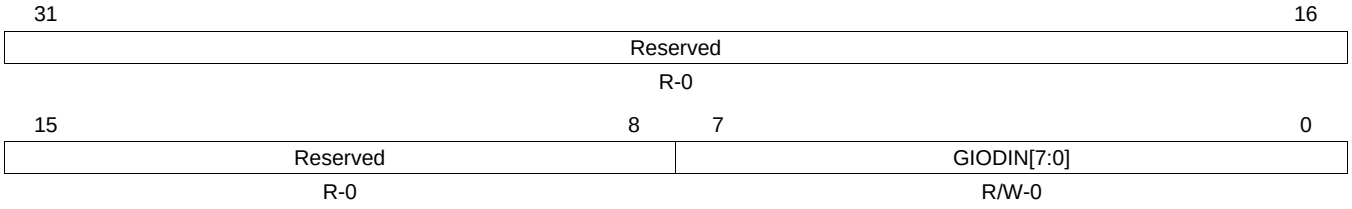
Table 19-14. GIO Data Direction Registers (GIODIR[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODIR[*n*]		GIO data direction of port n, pins [7:0]
		0	The GIO pin is an input. Note: If the pin direction is set as an input, the output buffer is tristated.
		1	The GIO pin is an output.

19.5.12 GIO Data Input Registers (GIODIN[A-B])

Values in the GIODIN register reflect the current state (high = 1 or low = 0) on the pins of the port. Figure 19-18 and Table 19-15 describe this register.

Figure 19-18. GIO Data Input Registers (GIODIN[A-B]) [offset = 38h, 58h]



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-15. GIO Data Input Registers (GIODIN[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODIN[*n*]		GIO data input for port n, pins [7:0]
		0	The pin is at logic low (0).
		1	The pin is at logic high (1).

19.5.13 GIO Data Output Registers (GIODOUT[A-B])

Values in the GIODOUT register specify the output state (high = 1 or low = 0) of the pins of the port when they are configured as outputs. Figure 19-19 and Table 19-16 describe this register.

NOTE: Values in the GIODSET register set the data output control register bits to 1 regardless of the current value in the GIODOUT bits.

Figure 19-19. GIO Data Output Registers (GIODOUT[A-B]) [offset = 3Ch, 5Ch]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIODOUT[7:0]	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-16. GIO Data Output Registers (GIODOUT[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODOUT[*n*]		GIO data output of port n, pins[7:0].
		0	The pin is driven to logic low (0).
		1	The pin is driven to logic high (1).
			Note: Output is in high impedance state if the GIOPDRx bit = 1 and GIODOUTx bit = 1.
			Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.

19.5.14 GIO Data Set Registers (GIODSET[A-B])

Values in this register set the data output control register bits to 1 regardless of the current value in the GIODOUT bits. The contents of this register reflect the contents of GIODOUT. Figure 19-20 and Table 19-17 describe this register.

Figure 19-20. GIO Data Set Registers (GIODSET[A-B]) [offset = 40h, 60h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIODSET[7:0]	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-17. GIO Data Set Registers (GIODSET[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODSET[*n*]		GIO data set for port n, pins[7:0]. This bit drives the output of GIO pin high.
		0	Write: Writing a 0 has no effect.
		1	Write: The corresponding GIO pin is driven to logic high (1).
			Note: The current logic state of the GIODOUT bit will also be displayed by this bit.
			Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.

19.5.15 GIO Data Clear Registers (GIODCLR[A-B])

Values in this register clear the data output register (GIO Data Output Register [A-H]) bit to 0 regardless of its current value. The contents of this register reflect the contents of GIODOUT. Figure 19-21 and Table 19-18 describe this register.

Figure 19-21. GIO Data Clear Registers (GIODCLR[A-B]) [offset = 44h, 64h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIODCLR[7:0]	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-18. GIO Data Clear Registers (GIODCLR[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODCLR[*n*]		GIO data clear for port n, pins[7:0]. This bit drives the output of GIO pin low.
		0	Write: Writing a 0 has no effect.
		1	Write: The corresponding GIO pin is driven to logic low (0).
			Note: The current logic state of the GIODOUT bit will also be displayed by this bit.
			Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.

19.5.16 GIO Open Drain Registers (GIOPDR[A-B])

Values in this register enable or disable the open drain capability of the data pins. Figure 19-22 and Table 19-19 describe this register.

Figure 19-22. GIO Open Drain Registers (GIOPDR[A-B]) [offset = 48h, 68h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIOPDR[7:0]	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-19. GIO Open Drain Registers (GIOPDR[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPDR[*n*]		GIO open drain for port n, pins[7:0]
		0	The GIO pin is configured in push/pull (normal GIO) mode. The output voltage is V_{OL} or lower if GIODOUT bit = 0 and V_{OH} or higher if GIODOUT bit = 1.
		1	The GIO pin is configured in open drain mode. The GIODOUTx bit controls the state of the GIO output buffer: GIODOUTx = 0, the GIO output buffer is driven low; GIODOUTx = 1, the GIO output buffer is tristated.

19.5.17 GIO Pull Disable Registers (GIOPULDIS[A-B])

Values in this register enable or disable the pull control capability of the pins. Figure 19-23 and Table 19-20 describe this register.

Figure 19-23. GIO Pull Disable Registers (GIOPULDIS[A-B]) [offset = 4Ch, 6Ch]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIOPULDIS[7:0]	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-20. GIO Pull Disable Registers (GIOPULDIS[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPULDIS[*n*]		GIO pull disable for port n, pins[7:0]. Writes to this bit will only take effect when the GIO pin configured as an input pin.
		0	The pull functionality is enabled.
		1	The pull functionality is disabled.
			Note: The GIO pin is placed in input mode by clearing the GIODIRx bit to 0.

19.5.18 GIO Pull Select Registers (GIOPSL[A-B])

Values in this register select the pull up or pull down functionality of the pins. Figure 19-24 and Table 19-21 describe this register.

Figure 19-24. GIO Pull Select Registers (GIOPSL[A-B]) [offset = 50h, 70h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIOPSL[7:0]	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19-21. GIO Pull Select Registers (GIOPSL[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPSL[*n*]		GIO pull select for port n, pins[7:0]
		0	The pull down functionality is select, when pull up/pull down logic is enabled.
		1	The pull up functionality is select, when pull up/pull down logic is enabled.
			Note: The pull up/pull down functionality is enabled by clearing corresponding bit in GIOPULDIS to 0.

19.6 I/O Control Summary

The behavior of the output buffer and the pull control is summarized in Table 19-22.

Table 19-22. Output Buffer and Pull Control Behavior for GIO Pins

Module under Reset?	Pin Direction (GIODIR)[1][2]	Open Drain Enable (GIOPDR)[1][3]	Pull Disable (GIOPULDIS)[1][4]	Pull Select (GIOPSL)[1][5]	Pull Control	Output Buffer[6]
Yes	X	X	X	X	Enabled	Disabled
No	0	X	0	0	Pull down	Disabled
No	0	X	0	1	Pull up	Disabled
No	0	X	1	0	Disabled	Disabled
No	0	X	1	1	Disabled	Disabled
No	1	0	X	X	Disabled	Enabled
No	1	1	X	X	Disabled	Enabled

(1) X = Don't care
(2) GIODIR = 0 for input; = 1 for output
(3) See Section 19.5.16
(4) GIOPULDIS = 0 for enabling pull control; = 1 for disabling pull control
(5) GIOPSL= 0 for pull-down functionality; = 1 for pull-up functionality
(6) If open drain is enabled, output buffer will be disabled if a high level (1) is being output.

Controller Area Network (DCAN) Module

This chapter describes the controller area network (DCAN) module.

Topic **Page**

20.1 Overview

The Controller Area Network is a high-integrity, serial, multi-master communication protocol for distributed real-time applications. This CAN module is implemented according to ISO 11898-1 and is suitable for industrial, automotive and general embedded communications.

20.1.1 Features

The DCAN module provides the following features:

Protocol

- Supports CAN protocol version 2.0 part A, B

Speed

- Bit rates up to 1 MBit/s

MailBox

- Configurable Message objects
- Individual identifier masks for each message object
- Programmable FIFO mode for message objects

Power

- Global power down and wakeup support
- Local power down and wakeup support

Debug

- Suspend mode for debug support
- Programmable loop-back modes for self-test operation
- Direct access to Message RAM in test mode
- Supports Two interrupt lines - Level 0 and Level 1

Others

- Automatic Message RAM initialization
- Automatic bus on after Bus-Off state by a programmable 32-bit timer
- CAN Rx / Tx pins configurable as general purpose IO pins
- Software module reset
- Message RAM parity check mechanism
- Dual clock source to reduce jitter

20.2 Module Operation

20.2.1 Functional Description

The CAN protocol is an ISO standard (ISO 11898) for serial data communication. This protocol uses Non-Return To Zero (NRZ) with bit-stuffing. And the communication is carried over a two-wire balanced signaling scheme.

The DCAN data communication happens through the CAN_TX and CAN_RX pins. An additional transceiver hardware is required for the connection to the physical layer (CAN bus) CAN_High and CAN_Low.

The DCAN register set can be accessed directly by the CPU. These registers are used to control and configure the CAN module and the Message RAM.

Individual CAN message objects should be configured for communication over a CAN network. The message objects and identifier masks are stored in the Message RAM.

The CAN module internally handles functions such as acceptance filtering, transfer of messages from and to the Message RAM, handling of transmission requests as well as the generation of interrupts.

20.2.2 CAN Blocks

The DCAN Module, shown in Figure 20-1, is comprised of the following basic blocks.

20.2.2.1 CAN Core

The CAN Core consists of the CAN Protocol Controller and the Rx/Tx Shift Register. It handles all ISO 11898-1 protocol functions.

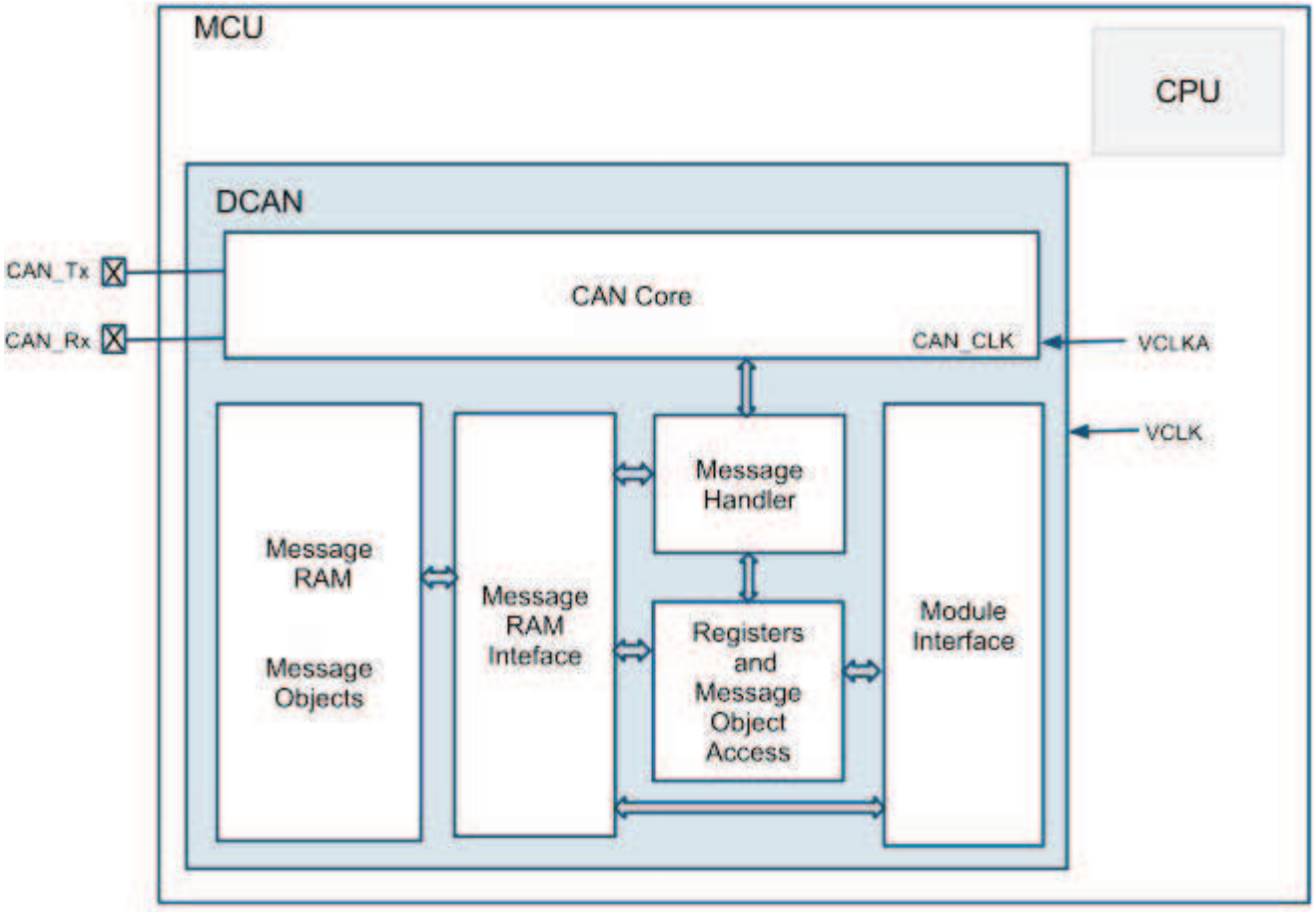
20.2.2.2 Message RAM

The DCAN Message RAM enables storage of CAN messages.

20.2.2.3 Message Handler

The Message Handler is a state machine that controls the data transfer between the single ported Message RAM and the CAN Core's Rx/Tx Shift Register. It also handles acceptance filtering and the interrupt request generation as programmed in the control registers.

Figure 20-1. Block Diagram



20.2.2.4 Message RAM Interface

The Interface Register sets control the CPU read and write accesses to the Message RAM.

There are three interface registers IF1,IF2 and IF3.

IF1 and IF2 Interface Registers sets for read and write access.

IF3 Interface Register set for read access only.

The Interface Registers have the same word-length as the Message RAM.

Additional information can be found in Section 20.2.6.

20.2.2.5 Register and Message Object Access

During normal operation, data consistency of the message objects is guaranteed by indirectly accessing the message objects through the interface registers IF1 and IF2.

In order to be able to perform tests on the message object memory, a dedicated test mode has been implemented, which allows direct access by the CPU. During normal operation direct access has to be avoided.

20.2.2.6 Dual Clock Source

Two clock domains are provided to the DCAN module:

1. VCLK - The peripheral synchronous clock domain as the general module clock source.
2. VCLKA - The peripheral asynchronous clock source domain provided to the CAN core as clock source (CAN_CLK) for generating the CAN Bit Timing.

If a frequency modulated clock output from FMPLL is used as the VCLK source, then VCLKA should be derived from an unmodulated clock source (for example, OSCIN source).

The clock source for VCLKA is selected by the Peripheral Asynchronous Clock Source Register in the system module.

Both clock domains can be derived from the same clock source (so that VCLK = VCLKA). However, if frequency modulation in the FMPLL is enabled (spread spectrum clock), then due to the high precision clocking requirements of the CAN Core, the FMPLL clock source should not be used for VCLKA. Alternatively, a separate clock without any modulation (for example, derived directly from the OSCIN clock) should be used for VCLKA.

Please refer to the system module reference guide and the device datasheet for more information how to configure the relevant clock source registers in the system module.

Between the two clock domains, a synchronization mechanism is implemented in the DCAN module in order to ensure correct data transfer.

NOTE: If the dual clock functionality is used, then VCLK must always be higher or equal to CAN_CLK (derived from the asynchronous clock source), in order to achieve a stable functionality of the DCAN. Here also the frequency shift of the modulated VCLK has to be considered:

$$f_{0,\ VCLK} \pm \Delta f_{FM,VCLK} \geq f_{CANCLK}$$

The CAN Core has to be programmed to at least 8 clock cycles per bit time. To achieve a transfer rate of 1 MBaud when using the asynchronous clock domain as the clock source for CAN_CLK, an oscillator frequency of 8MHz or higher has to be used.

20.2.3 CAN Bit Timing

The DCAN supports bit rates between less than 1 kBit/s and 1000 kBit/s.

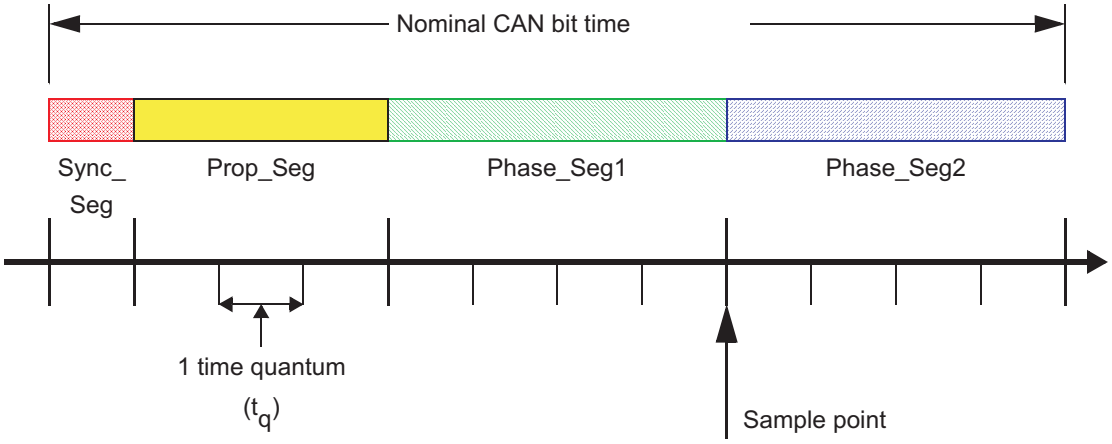
Each member of the CAN network has its own clock generator, typically derived from a crystal oscillator. The Bit timing parameters can be configured individually for each CAN node, creating a common Bit rate even though the CAN nodes' oscillator periods (f_{osc}) may be different.

20.2.3.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 20-2):

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

Figure 20-2. Bit Timing



Each segment consists of a specific number of time quanta. The length of one time quantum, (t_q), which is the basic time unit of the bit time, is given by the CAN_CLK and the Baud Rate Prescalers (BRPE and BRP). With these two Baud Rate Prescalers combined, divider values from 1 to 1024 can be programmed:

t_q = Baud Rate Prescaler / CAN_CLK

Apart from the fixed length of the synchronization segment, these numbers are programmable.

Table 20-1 describes the minimum programmable ranges required by the CAN protocol. A given bit rate may be met by different Bit time configurations.

NOTE: For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

Table 20-1. Parameters of the CAN Bit Time

Parameter	Range	Remark
Sync_Seg	1 t_q (fixed)	Synchronization of bus input to CAN_CLK
Prop_Seg	[1 … 8] t_q	Compensates for the physical delay times
Phase_Seg1	[1 … 8] t_q	May be lengthened temporarily by synchronization
Phase_Seg2	[1 … 8] t_q	May be shortened temporarily by synchronization
Synchronization Jump Width (SJW)	[1 … 4] t_q	May not be longer than either Phase Buffer Segment

20.2.3.1.1 Synchronization Segment

The Synchronization Segment (Sync_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync_Seg, its distance to the Sync_Seg is called the phase error of this edge.

20.2.3.1.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the CAN network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

20.2.3.1.3 Phase Buffer Segments and Synchronization

The Phase Buffer Segments (Phase_Seg1 and Phase_Seg2) and the Synchronization Jump Width (SJW) are used to compensate for the oscillator tolerance.

The Phase Buffer Segments surround the sample point. The Phase Buffer Segments may be lengthened or shortened by synchronization.

The Synchronization Jump Width (SJW) defines how far the resynchronizing mechanism may move the sample point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

Synchronizations occur on edges from recessive to dominant. Their purpose is to control the distance between edges and sample points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous sample point. A synchronization may be done only if a recessive bit was sampled at the previous sample point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise its distance to the Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

20.2.3.1.4 Oscillator Tolerance Range

With the introduction of CAN protocol version 1.2, the option to synchronize on edges from dominant to recessive became obsolete. Only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator's frequency f_{osc} around the nominal frequency f_{nom} with:

$$(1-df) \times f_{nom} \le f_{osc} \le (1+df) \times f_{nom} \quad (34)$$

depends on the proportions of Phase_Seg1, Phase_Seg2, SJW, and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met):

$$\text{I:} \quad df \le \frac{\min(\text{Phase\_Seg1}, \text{Phase\_Seg2})}{[2 \times (13 \times \text{bit\_time} - \text{Phase\_Seg2})]}$$

$$\text{II:} \quad df \le \frac{SJW}{20 \times \text{bit\_time}} \quad (35)$$

It has to be considered that SJW may not be larger than the smaller of the Phase Buffer Segments and that the Propagation Time Segment limits that part of the bit time that may be used for the Phase Buffer Segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8 μs) with a bus length of 40 m.

20.2.3.2 DCAN Bit Timing Registers

In the DCAN, the bit timing configuration is programmed in two register bytes, additionally a third byte for a baud rate prescaler extension of 4 bits (BREP) is provided. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte

In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1… n], values in the range of [0 … n-1] are programmed. That way, SJW (functional range of [1 … 4]) is represented by only two bits.

Therefore the length of the Bit time is (programmed values) [TSEG1 + TSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

The data in the Bit Timing Register (BTR) is the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRPE/BRP) defines the length of the time quantum (the basic time unit of the bit time); the Bit Timing Logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

20.2.3.2.1 Calculation of the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting Bit time (1 / Bit rate) must be an integer multiple of the CAN clock period.

NOTE: 8 MHz is the minimum CAN clock frequency required to operate the DCAN at a bit rate of 1 MBit/s.

The bit time may consist of 8 to 25 time quanta. The length of the time quantum t_q is defined by the Baud Rate Prescaler with t_q = (Baud Rate Prescaler) / CAN_CLK. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The Sync_Seg is 1 t_q long (fixed), leaving (bit time – Prop_Seg – 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than any CAN controller's Information Processing Time in the network, which is device dependent and can be in the range of [0 … 2] t_q.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration that allows the highest oscillator tolerance range.

CAN nodes with different clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

The resulting configuration is written into the Bit Timing Register:

Tseg2 = Phase_Seg2-1
Tseg1 = Phase_Seg1+ Prop_Seg-1
SJW = SynchronizationJumpWidth-1
BRP = Prescaler-1

20.2.3.2.2 Example for Bit Timing at High Baudrate

In this example, the frequency of CAN_CLK is 10 MHz, BRP is 0, the bit rate is 1 MBit/s.

t_q	100	ns	=	t_{CAN_CLK}
delay of bus driver	60	ns		
delay of receiver circuit	40	ns		
delay of bus line (40m)	220	ns		
t_{Prop}	700	ns	=	INT (2 × delays + 1) = 7 × t_q
t_{SJW}	100	ns	=	1 × t_q
t_{TSeg1}	800	ns	=	t_{Prop} + t_{SJW}
t_{TSeg2}	100	ns	=	Information Processing Time + 1 × t_q
$t_{Sync\text{-}Seg}$	100	ns	=	1 × t_q
bit time	1000	ns	=	$t_{Sync\text{-}Seg}$ + t_{TSeg1} + t_{TSeg2}
tolerance for CAN_CLK	1.58	%	=	$\frac{\min(TSeg1, TSeg2)}{[2 \times (13 \times bit\_time - TSeg2)]}$ (36)
			=	$\frac{0.1\,\mu s}{[2 \times (13 \times 1\,\mu s - 0.1\,\mu s)]}$ (37)
			=	0.38%

In this example, the concatenated bit time parameters are $(1\text{-}1)_3\&(8\text{-}1)_4\&(1\text{-}1)_2\&(1\text{-}1)_6$, so the Bit Timing Register is programmed to 0000 0700h.

20.2.3.2.3 Example for Bit Timing at Low Baudrate

In this example, the frequency of CAN_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

t_q	1	µs	=	2 × t_{CAN_CLK}
delay of bus driver	200	ns		
delay of receiver circuit	80	ns		
delay of bus line (40m)	220	ns		
t_{Prop}	1	µs	=	1 × t_q
t_{SJW}	4	µs	=	4 × t_q
t_{TSeg1}	5	µs	=	t_{Prop} + t_{SJW}
t_{TSeg2}	3	µs	=	Information Processing Time + 3 × t_q
$t_{Sync\text{-}Seg}$	1	µs	=	1 × t_q
bit time	9	µs	=	$t_{Sync\text{-}Seg}$ + t_{TSeg1} + t_{TSeg2}
tolerance for CAN_CLK	0.43	%	=	$\frac{\min(TSeg1, TSeg2)}{[2 \times (13 \times bit\_time - TSeg2)]}$ (38)
			=	$\frac{3\mu s}{[2 \times (13 \times 9\mu s - 3\mu s)]}$ (39)
			=	1.32%

In this example, the concatenated bit time parameters are $(3\text{-}1)_3\&(5\text{-}1)_4\&(4\text{-}1)_2\&(2\text{-}1)_6$, so the Bit Timing Register is programmed to 0000 24C1h.

20.2.4 CAN Module Configuration

After a hardware reset all CAN protocol functions are disabled. The CAN module must be initialized and configured before it can participate on the CAN bus.

20.2.4.1 DCAN RAM Initialization through harware

To start with a clean DCAN RAM ,the complete DCAN RAM can be initialized with zeros and the ecc/parity bits set accordingly by configuring the following registers in the system module:

1. Memory Hardware Initialization Global Control Register (MINITGCR)
2. Memory Initialization Enable Register (MSINENA)

For more details on RAM hardware initialization support, refer to the system module.

20.2.4.2 CAN Module Initialization

To initialize the CAN Controller, you have to set up the CAN Bit timing and those message objects that have to be used for CAN communication. Message objects that are not needed, can be deactivated.

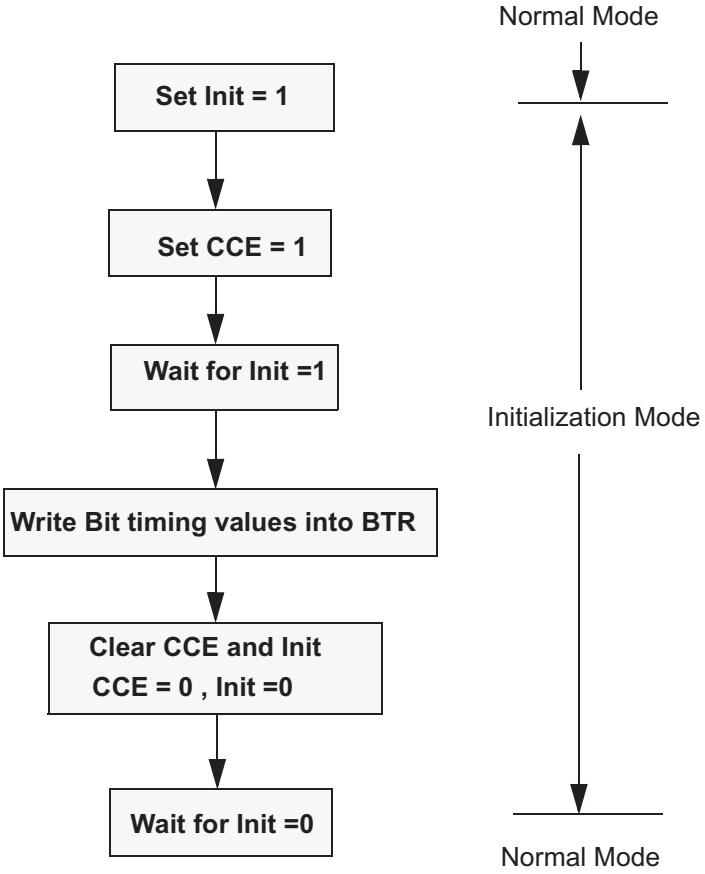
So the two critical steps are:

1. Configuration of CAN Bit Timings
2. Configuration of Message Objects

20.2.4.2.1 Software Configuration of CAN Bit Timings

This step involves configuring the CAN baud rate register with the calculated CAN bit timing value. The calculation procedure of CAN bit timing values for BTR register are mentioned in Section 20.2.3. Refer to Figure 20-3 for CAN bit timing software configuration flow.

Figure 20-3. CAN Bit-timing Configuration



Step 1: Enter “initialization mode” by setting the Init (Initialization) bit in the CAN Control Register.

While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN_TX output is recessive (high).

The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

Also note that the CAN module is also in initialization mode on hardware reset and during Bus-Off.

Step 2: Set the CCE (Configure Change Enable) bit in the CAN Control Register.

The access to the Bit Timing Register for the configuration of the Bit timing is enabled when both Init and CCE bits in the CAN Control Register are set.

Step 3: Wait for the Init bit to get set. This would make sure that the module has entered “Initialization mode”.

Step 4: Write the Bit-Timing values into the BTR (Bit-Timing Register) Register.

Refer to Section 20.2.3.2.1 for BTR value calculation for a given bit-timing.

Step 5: Clear the CCE bit followed by Init bit.

Step 6: Wait for the Init bit to clear. This would make sure that the module has come out of “initialization mode”.

After step 6 (Init bit cleared), the module will attempt a synchronization on the CAN bus, provided that the BTR settings are meeting the CAN bus parameters.

NOTE: The module would not come out of the “initialization mode” if any incorrect BTR values are written in step 4.

20.2.4.2.2 Configuration of Message Objects

The whole Message RAM should be configured before putting the CAN into operation. All the message objects are deactivated by default. The user should configure the message object that are to be used to a particular identifier. The user can change the configuration of any message object or deactivate it when required.

The message objects can be configured only through the Interface registers (IFx) and the CPU does not have direct access to the message object (Message RAM) when DCAN is in operation.

To configure the message objects, the user must know about:

1. The message object structure (Section 20.2.5)
2. The interface register set (IFx) (Section 20.2.6)

NOTE: The message objects initialization is independent of the bit-timing configuration procedure.

20.2.5 Message RAM

The DCAN Message RAM contains message objects and parity bits for the message objects.

20.2.5.1 Structure of Message Objects

Figure 20-4 shows the structure of a message object.

The grayed fields are those parts of the message object that are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

Figure 20-4. Structure of a Message Object

Message Object												
UMask	Msk[28:0]	MXtd	MDir	EoB	unused	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID[28:0]	Xtd	Dir	DLC[3:0]	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7

Table 20-2. Message Object Field Descriptions

Name	Value	Description
MsgVal		Message valid
	0	The message object is ignored by the Message Handler.
	1	The message object is to be used by the Message Handler.
		Note: The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation see Section 20.2.7.6 and Section 20.2.7.7.
UMask		Use Acceptance Mask
	0	Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering.
	1	Mask bits are used for acceptance filtering.
		Note: If the UMask bit is set to 1, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to 1.
ID[28:0]		Message Identifier
	ID[28:0]	29-bit ("extended") identifier bits
	ID[28:18]	11-bit ("standard") identifier bits
Msk[28:0]		Identifier Mask
	0	The corresponding bit in the message identifier is not used for acceptance filtering (don't care).
	1	The corresponding bit in the message identifier is used for acceptance filtering.
Xtd		Extended Identifier
	0	The 11-bit ("standard") identifier will be used for this message object.
	1	The 29-bit ("extended") identifier will be used for this message object.
MXtd		Mask Extended Identifier
	0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
	1	The extended identifier bit (IDE) is used for acceptance filtering.
		Note: When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
Dir		Message Direction
	0	Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message is stored in this message object.
	1	Direction = transmit: On TxRqst, a data frame is transmitted. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
MDir		Mask Message Direction
	0	The message direction bit (Dir) has no effect on the acceptance filtering.
	1	The message direction bit (Dir) is used for acceptance filtering.

Table 20-2. Message Object Field Descriptions (continued)

Name	Value	Description
EOB		End of Block
	0	The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
	1	The message object is a single message object or the last message object in a FIFO Buffer Block.
		Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to 1.
NewDat		New Data
	0	No new data has been written into the data bytes of this message object by the Message Handler since the last time when this flag was cleared by the CPU.
	1	The Message Handler or the CPU has written new data into the data bytes of this message object.
MsgLst		Message Lost (only valid for message objects with direction = receive)
	0	No message was lost since the last time when this bit was reset by the CPU.
	1	The Message Handler stored a new message into this message object when NewDat was still set, so the previous message has been overwritten.
RxIE		Receive Interrupt Enable
	0	IntPnd will not be triggered after the successful reception of a frame.
	1	IntPnd will be triggered after the successful reception of a frame.
TxIE		Transmit Interrupt Enable
	0	IntPnd will not be triggered after the successful transmission of a frame.
	1	IntPnd will be triggered after the successful transmission of a frame.
IntPnd		Interrupt Pending
	0	This message object is not the source of an interrupt.
	1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
RmtEn		Remote Enable
	0	At the reception of a Remote Frame, TxRqst is not changed.
	1	At the reception of a Remote Frame, TxRqst is set.
TxRqst		Transmit Request
	0	This message object is not waiting for a transmission.
	1	The transmission of this message object is requested and is not yet done.
DLC[3:0]		Data Length Code
	0-8	Data Frame has 0-8 data bytes.
	9-15	Data Frame has 8 data bytes.
		Note: The Data Length Code of a message object must be defined to the same value as in the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message.
Data 0		1st data byte of a CAN Data Frame
Data 1		2nd data byte of a CAN Data Frame
Data 2		3rd data byte of a CAN Data Frame
Data 3		4th data byte of a CAN Data Frame
Data 4		5th data byte of a CAN Data Frame
Data 5		6th data byte of a CAN Data Frame
Data 6		7th data byte of a CAN Data Frame
Data 7		8th data byte of a CAN Data Frame
		Note: Byte Data 0 is the first data byte shifted into the shift register of the CAN Core during a reception, byte Data 7 is the last. When the Message Handler stores a data frame, it will write all the eight data bytes into a message object. If the Data Length Code is less than 8, the remaining bytes of the message object may be overwritten by undefined values.

20.2.5.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:

Message RAM base address + (message object number) × 0x20.

This means that Message Object 1 starts at offset 0x0020; Message Object 2 starts at offset 0x0040, and so on.

NOTE: 0 is not a valid message object number. Writing to the address of an unimplemented message object may overwrite an implemented message object.

The base address for DCAN1 RAM is 0xFF1E 0000, and DCAN2 RAM is 0xFF1C 0000.

Message Object number 1 has the highest priority.

Table 20-3. Message RAM Addressing in Debug/Suspend and RDA mode

Message Object Number	Base Address Offset	Word Number	Debug/Suspend mode, see Section 20.2.5.3	RDA mode, see Section 20.2.5.4
1	0x0020	1	Parity	Data Bytes 4-7
	0x0024	2	MXtd, MDir, Mask	Data Bytes 0-3
	0x0028	3	Xtd, Dir, ID	ID[27:0], DLC
	0x002C	4	Ctrl	Mask, Xtd, Dir, ID[28]
	0x0030	5	Data Bytes 3-0	Parity, Ctrl, MXtd, MDir
	0x0034	6	Data Bytes 7-4	--
:	:	:	:	:
31	0x03E0	1	Parity	Data Bytes 4-7
	0x03E4	2	MXtd, MDir, Mask	Data Bytes 0-3
	0x03E8	3	Xtd, Dir, ID	ID[27]:0, DLC
	0x03EC	4	Ctrl	Mask, Xtd, Dir, ID[28]
	0x03F0	5	Data Bytes 3-0	Parity, Ctrl, MXtd, MDir
	0x03F4	6	Data Bytes 7-4	--
:	:	:	:	:
63	0x07E0	1	Parity	Data Bytes 4-7
	0x07E4	2	MXtd, MDir, Mask	Data Bytes 0-3
	0x07E8	3	Xtd, Dir, ID	ID[27:0], DLC
	0x07EC	4	Ctrl	Mask, Xtd, Dir, ID[28]
	0x07F0	5	Data Bytes 3-0	Parity, Ctrl, MXtd, MDir
	0x07F4	6	Data Bytes 7-4	--
64	0x0000	1	Parity	Data Bytes 4-7
	0x0004	2	MXtd, MDir, Mask	Data Bytes 0-3
	0x0008	3	Xtd, Dir, ID	ID[27]:0, DLC
	0x000C	4	Ctrl	Mask, Xtd, Dir, ID[28]
	0x0010	5	Data Bytes 3-0	Parity, Ctrl, MXtd, MDir
	0x0014	6	Data Bytes 7-4	--

20.2.5.3 Message RAM Representation in Debug/Suspend Mode

In Debug/Suspend mode, the Message RAM will be memory mapped. This allows the external debug unit to access the Message RAM.

NOTE: During Debug/Suspend Mode, the Message RAM cannot be accessed via the IFx register sets.

Figure 20-5. Message RAM Representation in Debug/Suspend Mode

Bit	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
MsgAddr + 0x00	Reserved															
	Reserved															Parity[4:0]
MsgAddr + 0x04	MXtd	MDir	Rsvd	Msk[28:16]												
	Msk[15:0]															
MsgAddr + 0x08	Rsvd	Xtd	Dir	ID[28:16]												
	ID[15:0]															
MsgAddr + 0x0C	Reserved															
	Rsvd	MsgLst	Rsvd	UMask	TxIE	RxIE	RmtEn	Rsvd	EOB	Reserved			DLC[3:0]			
MsgAddr + 0x10	Data 3								Data 2							
	Data 1								Data 0							
MsgAddr + 0x14	Data 7								Data 6							
	Data 5								Data 4							

20.2.5.4 Message RAM Representation in Direct Access Mode

When the RDA bit in Test Register is set while the DCAN module is in Test Mode (Test bit in CAN control register is set), the CPU has direct access to the Message RAM. Due to the 32-bit bus structure, the RAM is split into word lines to support this feature. The CPU has access to one word line at a time only.

In RAM Direct Access mode, the RAM is represented by a continuous memory space within the address frame of the DCAN module, starting at the Message RAM base address.

NOTE: During Direct Access Mode, the Message RAM cannot be accessed via the IFx register sets. Before entering RDA mode, it must be ensured that the Init bit is set to avoid any conflicts with the message handler accessing the message RAM.

Any read or write to the RAM addresses for RamDirectAccess during normal operation mode (TestMode bit or RDA bit is not set) will be ignored.

Writes to Reserved bits have no effect.

Figure 20-6. Message RAM Representation in RAM Direct Access Mode

Bit	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
MsgAddr + 0x00	Data 4								Data 5							
	Data 6								Data 7							
MsgAddr + 0x04	Data 0								Data 1							
	Data 2								Data 3							
MsgAddr + 0x08	ID[27:12]															
	ID[11:0]												DLC[3:0]			
MsgAddr + 0x0C	Msk[28:13]															
	Msk[12:0]													Xtd	Dir	ID[28]
MsgAddr + 0x10	Reserved															
	Parity[3:0]				Reserved				MsgLst	UMask	TxIE	RxIE	RmtEn	EOB	MXtd	MDir

20.2.6 Message Interface Register Sets

Accesses to the Message RAM are performed via the Interface Register sets.

1. Interface Register 1 and 2
2. Interface Register 3

The third Interface Register set IF3 can be configured to automatically receive control and user data from the Message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from Message RAM to IF3 Register set.

The Message Handler avoids potential conflicts between concurrent accesses to Message RAM and CAN frame reception/transmission.

There are two modes where the Message RAM can be directly accessed by the CPU:

1. In Debug/Suspend mode (see Section 20.2.5.3)
2. In RAM Direct Access (RDA) mode (see Section 20.2.5.4)

For the Message RAM Base address, please refer to the device datasheet.

A complete message object (see Section 20.2.5.1) or parts of the message object may be transferred between the Message RAM and the IF1/IF2 Register set (see Section 20.3.20) in one single transfer.

20.2.6.1 Message Interface Register Sets 1 and 2

The Interface Register sets IF1 and IF2 provide indirect read/write access from the CPU to the Message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

Table 20-4. Message Interface Register Sets 1 and 2

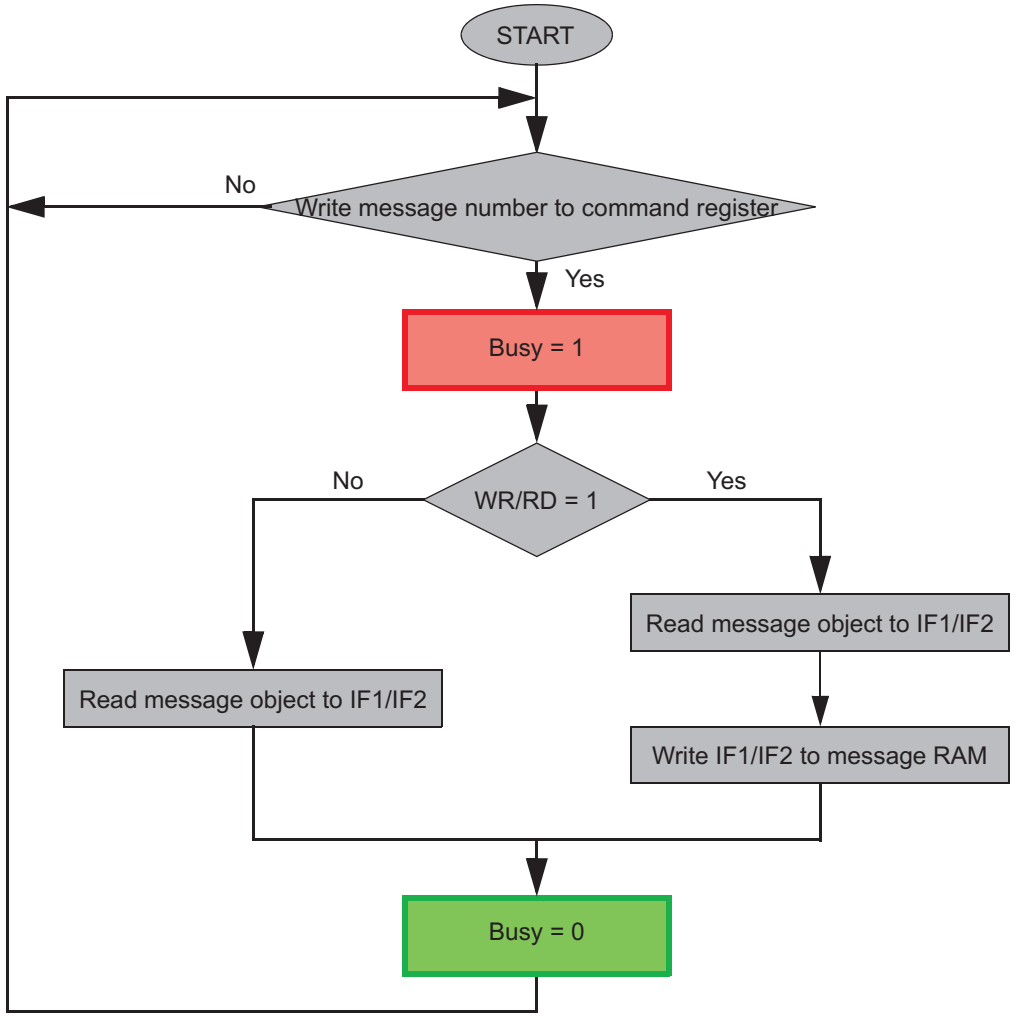
Address	IF1 Register Set		Address	IF2 Register Set	
[CAN Base +]	31 16	15 0	[CAN Base +]	31 16	15 0
0x100	IF1 Command Mask	IF1 Command Request	0x120	IF2 Command Mask	IF2 Command Request
0x104	IF1 Mask 2	IF1 Mask 1	0x124	IF2 Mask 2	IF2 Mask 1
0x108	IF1 Arbitration 2	IF1 Arbitration 1	0x128	IF2 Arbitration 2	IF2 Arbitration 1
0x10C	Rsvd	IF1 Message Control	0x12C	Rsvd	IF2 Message Control
0x110	IF1 Data A 2	IF1 Data A 1	0x130	IF2 Data A 2	IF2 Data A 1
0x114	IF1 Data B 2	IF1 Data B 1	0x134	IF2 Data B 2	IF2 Data B 1

20.2.6.2 Using Message Interface Register Sets 1 and 2

The Command Register addresses the desired message object in the Message RAM and specifies whether a complete message object or only parts should be transferred. The data transfer is initiated by writing the message number to the bits [7:0] of the Command Register.

When the CPU initiates a data transfer between the IF1/IF2 Registers and Message RAM, the Message Handler sets the Busy bit in the respective Command Register to '1'. After the transfer has completed, the Busy bit is set back to '0' (see Figure 20-7).

Figure 20-7. Data Transfer Between IF1 / IF2 Registers and Message RAM



20.2.6.3 Message Interface Register 3

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU.

Table 20-5. Message Interface Register 3

Address	IF3 Register Set	
[CAN Base +]	31 16	15 0
0x140	reserved	IF3 Observation
0x144	IF3 Mask 2	IF3 Mask 1
0x148	IF3 Arbitration 2	IF3 Arbitration 1
0x14C	reserved	IF3 Message Control
0x150	IF3 Data A 2	IF3 Data A 1
0x154	IF3 Data B 2	IF3 Data B 1
...		
0x160	IF3 Update Enable 2	IF3 Update Enable 1
0x164	IF3 Update Enable 4	IF3 Update Enable 3
0x168	IF3 Update Enable 6	IF3 Update Enable 5
0x16C	IF3 Update Enable 8	IF3 Update Enable 7

The automatic update functionality can be programmed for each message object (see IF3 Update Enable Register, Section 20.3.29).

All valid message objects in Message RAM that are configured for automatic update, will be checked for active NewDat flags. If such a message object is found, it will be transferred to the IF3 register, as controlled by IF3 Observation register. If more than one NewDat flag is active, the message object with the lowest number has the highest priority for automatic IF3 update.

The NewDat bit in the message object will be reset by a transfer to IF3.

NOTE: The IF3 register set can not be used for transferring data into message objects.

20.2.7 Message Object Configurations

This section describes the possible message object configurations for CAN communication.

20.2.7.1 Configuration of a Transmit Object for Data Frames

Figure 20-8 shows how a Transmit Object can be initialized.

Figure 20-8. Initialization of a Transmit Object

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

The Arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (Standard Frame) is used (Xtd = 0), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored.

The Data Registers (DLC[3:0] and Data0-7) are given by the application, TxRqst and RmtEn should not be set before the data is valid.

If the TxIE bit is set, the IntPnd bit will be set after a successful transmission of the message object.

If the RmtEn bit is set, a matching received Remote Frame will cause the TxRqst bit to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask = 1) to allow groups of Remote Frames with similar identifiers to set the TxRqst bit. The Dir bit should not be masked. For details, see Section 20.2.8.8. Identifier masking must be disabled (UMask = 0) if no Remote Frames are allowed to set the TxRqst bit (RmtEn = 0).

20.2.7.2 Configuration of a Transmit Object for Remote Frames

It is not necessary to configure Transmit Objects for the transmission of Remote Frames. Setting TxRqst for a Receive Object will cause the transmission of a Remote Frame with the same identifier as the Data Frame for which this receive Object is configured.

20.2.7.3 Configuration of a Single Receive Object for Data Frames

Figure 20-9 shows how a Receive Object for Data Frames can be initialized.

Figure 20-9. Initialization of a Single Receive Object for Data Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

The Arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used (Xtd = 0), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a Data Frame with an 11-bit Identifier is received, ID[17:0] will be set to 0.

The Data Length Code (DLC[3:0]) is given by the application. When the Message Handler stores a Data Frame in the message object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the message object may be overwritten by non specified values.

The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask = 1) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. If some bits of the Mask bits are set to “don’t care”, the corresponding bits of the Arbitration Register will be overwritten by the bits of the stored Data Frame.

If the RxIE bit is set, the IntPnd bit will be set when a received Data Frame is accepted and stored in the message object.

If the TxRqst bit is set, the transmission of a Remote Frame with the same identifier as actually stored in the Arbitration bits will be triggered. The content of the Arbitration bits may change if the Mask bits are used (UMask = 1) for acceptance filtering.

20.2.7.4 Configuration of a Single Receive Object for Remote Frames

Figure 20-10 shows how a Receive Object for Remote Frames can be initialized.

Figure 20-10. Initialization of a Single Receive Object for Remote Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	appl.	0	0	0	0

Receive Objects for Remote Frames may be used to monitor Remote Frames on the CAN bus. The Remote Frame stored in the Receive Object will not trigger the transmission of a Data Frame. Receive Objects for Remote Frames may be expanded to a FIFO buffer, see Section 20.2.7.5.

UMask must be set to 1. The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be set to “must-match” or to “don’t care”, to allow groups of Remote Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. For details, see Section 20.2.8.8.

The Arbitration bits (ID[28:0] and Xtd bit) may be given by the application. They define the identifier and type of accepted received Remote Frames. If some bits of the Mask bits are set to “don’t care”, the corresponding bits of the Arbitration bits will be overwritten by the bits of the stored Remote Frame. If an 11-bit Identifier (Standard Frame) is used (Xtd = 0), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a Remote Frame with an 11-bit Identifier is received, ID[17:0] will be set to 0.

The Data Length Code (DLC[3:0]) may be given by the application. When the Message Handler stores a Remote Frame in the message object, it will store the received Data Length Code. The data bytes of the message object will remain unchanged.

If the RxIE bit is set, the IntPnd bit will be set when a received Remote Frame is accepted and stored in the message object.

20.2.7.5 Configuration of a FIFO Buffer

With the exception of the EoB bit, the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a single Receive Object.

To concatenate multiple message objects to a FIFO Buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest number will be the first message object of the FIFO Buffer. The EoB bit of all message objects of a FIFO Buffer except the last one have to be programmed to zero. The EoB bits of the last message object of a FIFO Buffer is set to 1, configuring it as the end of the block.

20.2.7.6 Reconfiguration of Message Objects for the Reception of Frames

A message object with Dir = 0 is configured for the reception of data frames, with Dir = 1 AND Umask = 1 AND RmtEn = 0 it is configured for the reception of remote frames.

It is necessary to reset MsgVal to not valid before changing any of the following configuration and control bits: ID[28:0], Xtd, Dir, DLC[3:0], RxIE, TxIE, RmtEn, EoB, Umask, Msk[28:0], MXtd, and MDir.

These parts of a message object may be changed without clearing MsgVal: Data[7:0], TxRqst, NewDat, MsgLst, and IntPnd.

20.2.7.7 Reconfiguration of Message Objects for the Transmission of Frames

A message object with Dir = 1 AND (Umask = 0 OR RmtEn = 1) is configured for the transmission of data frames.

It is necessary to reset MsgVal to not valid before changing any of the following configuration and control bits: Dir, RxIE, TxIE, RmtEn, EoB, Umask, Msk[28:0], MXtd, and MDir.

These parts of a message object may be changed without clearing MsgVal: ID[28-0], Xtd, DLC[3:0], Data[7:0], TxRqst, NewDat, MsgLst, and IntPnd.

20.2.8 Message Handling

When initialization is finished, the DCAN module synchronizes itself to the traffic on the CAN bus. It does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects.

The application has to update the data of the messages to be transmitted and enable and request their transmission. The transmission is requested automatically when a matching Remote Frame is received.

The application may read messages that are received and accepted. Messages that are not read before the next messages is accepted for the same message object will be overwritten.

Messages may be read based on interrupts or by polling.

20.2.8.1 Message Handler Overview

The Message Handler state machine controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM. It performs the following tasks:

- Data Transfer from Message RAM to CAN Core (messages to be transmitted).
- Data Transfer from CAN Core to the Message RAM (received messages).
- Data Transfer from CAN Core to the Acceptance Filtering unit.
- Scanning of Message RAM for a matching message object (acceptance filtering).
- Scanning the same message object after being changed by IF1/IF2 registers when priority is same or higher as message the object found by last scanning.
- Handling of TxRqst flags.
- Handling of interrupt flags.

The Message Handler registers contains status flags of all message objects grouped into the following topics:

- Transmission Request flags
- New Data flags
- Interrupt Pending Flags
- Message Valid Registers

Instead of collecting the listed status information of each message object via IFx registers separately, these Message Handler registers provides a fast and easy way to get an overview (for example, about all pending transmission requests).

All Message Handler registers are read-only.

20.2.8.2 Receive/Transmit Priority

The receive/transmit priority for the message objects is attached to the message number, not to the CAN identifier. Message object 1 has the highest priority, while the last implemented message object has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding message object, so messages with the highest priority can be placed in the message objects with the lowest numbers.

The acceptance filtering for received Data Frames or Remote Frames is also done in ascending order of message objects, so a frame that has been accepted by a message object cannot be accepted by another message object with a higher Message Number. The last message object may be configured to accept any Data Frame or Remote Frame that was not accepted by any other message object, for nodes that need to log the complete message traffic on the CAN bus.

20.2.8.3 Transmission of Messages in Event Driven CAN Communication

If the shift register of the CAN Core is ready for loading and if there is no data transfer between the IFx Registers and Message RAM, the d bits in the Message Valid Register and the TxRqst bits in the Transmission Request Register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The message object's NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = 0) since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the DCAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

If Automatic Retransmission mode is disabled by setting the DAR bit in the CAN Control Register, the behavior of bits TxRqst and NewDat in the Message Control Register of the Interface Register set is as follows:

- When a transmission starts, the TxRqst bit of the respective Interface Register set is reset, while bit NewDat remains set.
- When the transmission has been successfully completed, the NewDat bit is reset.

When a transmission failed (lost arbitration or error) bit NewDat remains set. To restart the transmission, the application has to set TxRqst again.

Received Remote Frames do not require a Receive Object. They will automatically trigger the transmission of a Data Frame, if in the matching Transmit Object the RmtEn bit is set.

20.2.8.4 Updating a Transmit Object

The CPU may update the data bytes of a Transmit Object any time via the IF1/IF2 Interface Registers, neither d nor TxRqst have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes in the corresponding IF1/IF2 Data A Register or IF1/IF2 Data B Register have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the IF1/IF2 Data Register or the message object is transferred to the IF1/IF2 Data Register before the CPU writes the new data bytes.

When only the data bytes are updated, first 0x87 can be written to bits [23:16] of the Command Register and then the number of the message object is written to bits [7:0] of the Command Register, concurrently updating the data bytes and setting TxRqst with NewDat.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat has to be set together with TxRqst in event driven CAN communication. For details, see Section 20.2.8.3.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

20.2.8.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the Transmit Objects may be managed dynamically. The CPU can write the whole message (Arbitration, Control, and Data) into the Interface Register. The bits [23:16] of the Command Register can be set to 0xB7 for the transfer of the whole message object content into the message object. Before changing the configuration of a message object, MsgVal has to be reset (see Section 20.2.7.7).

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however it will not be repeated if it is disturbed.

To only update the data bytes of a message to be transmitted, bits [23:16] of the Command Register should be set to 0x87.

NOTE: After the update of the Transmit Object, the Interface Register set will contain a copy of the actual contents of the object, including the part that had not been updated.

20.2.8.6 Acceptance Filtering of Received Messages

When the arbitration and control bits (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the shift register of the CAN Core, the Message Handler starts to scan of the Message RAM for a matching valid message object:

- The Acceptance Filtering unit is loaded with the arbitration bits from the CAN Core shift register.
- Then the arbitration and mask bits (including MsgVal, UMask, NewDat, and EoB) of Message Object 1 are loaded into the Acceptance Filtering unit and are compared with the arbitration bits from the shift register. This is repeated for all following message objects until a matching message object is found, or until the end of the Message RAM is reached.
- If a match occurs, the scanning is stopped and the Message Handler proceeds depending on the type of the frame (Data Frame or Remote Frame) received.

20.2.8.7 Reception of Data Frames

The Message Handler stores the message from the CAN Core shift register into the respective message object in the Message RAM. Not only the data bytes, but all arbitration bits and the Data Length Code are stored into the corresponding message object. This ensures that the data bytes stay associated to the identifier even if arbitration mask registers are used.

The NewDat bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset the NewDat bit when it reads the message object. If at the time of the reception the NewDat bit was already set, MsgLst is set to indicate that the previous data (supposedly not seen by the CPU) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the Interrupt Register to point to this message object.

The TxRqst bit of this message object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

20.2.8.8 Reception of Remote Frames

When a Remote Frame is received, three different configurations of the matching message object have to be considered:

1. Dir = 1 (direction = transmit), RmtEn = 1, UMask = 1 or 0. The TxRqst bit of this message object is set at the reception of a matching Remote Frame. The rest of the message object remains unchanged.
2. Dir = 1 (direction = transmit), RmtEn = 0, UMask = 0. The Remote Frame is ignored, this message object remains unchanged.
3. Dir = 1 (direction = transmit), RmtEn = 0, UMask = 1. The Remote Frame is treated similar to a received Data Frame. At the reception of a matching Remote Frame, the TxRqst bit of this message object is reset. The arbitration and control bits (Identifier + IDE + RTR + DLC) from the shift register are stored in the message object in the Message RAM and the NewDat bit of this message object is set. The data bytes of the message object remain unchanged.

20.2.8.9 Reading Received Messages

The CPU may read a received message any time via the IFx Interface Registers, the data consistency is guaranteed by the Message Handler state machine.

Typically the CPU will write first 0x7F to bits [23:16] and then the number of the message object to bits [7:0] of the Command Register. That combination will transfer the whole received message from the Message RAM into the Interface Register set. Additionally, the bits NewDat and IntPnd are cleared in the Message RAM (not in the Interface Register set). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

If the message object uses masks for acceptance filtering, the arbitration bits show which of the different matching messages has been received.

The actual value of NewDat shows whether a new message has been received since last time when this message object was read. The actual value of MsgLst shows whether more than one message have been received since the last time when this message object was read. MsgLst will not be automatically reset.

20.2.8.10 Requesting New Data for a Receive Object

By means of a Remote Frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object will cause the transmission of a Remote Frame with the receive object's identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TxRqst bit is automatically reset.

Setting the TxRqst bit without changing the contents of a message object requires the value 0x84 in bits [23:16] of the Command Register.

20.2.8.11 Storing Received Messages in FIFO Buffers

Several message objects may be grouped to form one or more FIFO Buffers. Each FIFO Buffer configured to store received messages with a particular (group of) Identifier(s). Arbitration and Mask Registers of the FIFO Buffer's message objects are identical. The EoB (End of Buffer) bits of all but the last of the FIFO Buffer's message objects are 0, in the last one the EoB bit is 1.

Received messages with identifiers matching to a FIFO Buffer are stored into a message object of this FIFO Buffer, starting with the message object with the lowest message number.

When a message is stored into a message object of a FIFO Buffer the NewDat bit of this message object is set. By setting NewDat while EoB is 0, the message object is locked for further write accesses by the Message Handler until the CPU has cleared the NewDat bit.

Messages are stored into a FIFO Buffer until the last message object of this FIFO Buffer is reached. If none of the preceding message objects is released by writing NewDat to 0, all further messages for this FIFO Buffer will be written into the last message object of the FIFO Buffer (EoB = 1) and therefore overwrite previous messages in this message object.

20.2.8.12 Reading from a FIFO Buffer

Several messages may be accumulated in a set of message objects that are concatenated to form a FIFO Buffer before the application program is required (in order to avoid the loss of data) to empty the buffer.

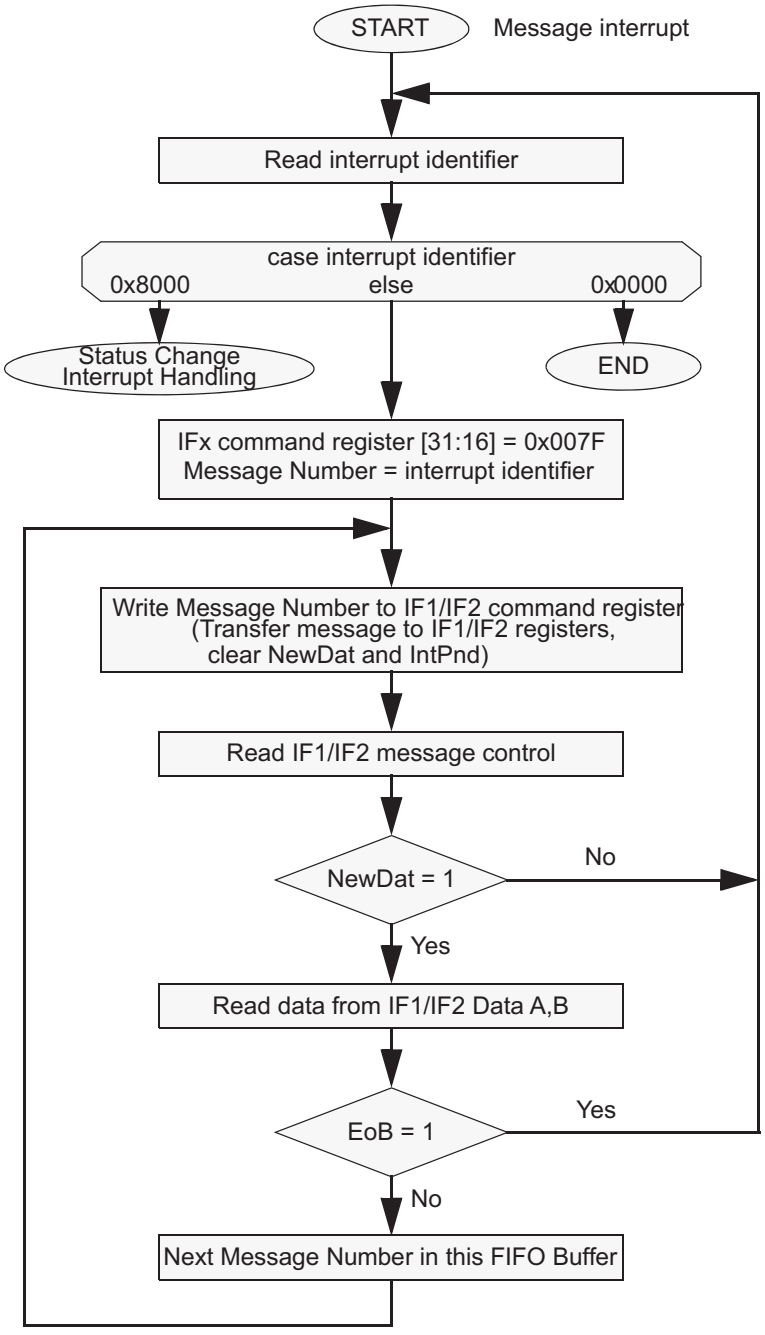
A FIFO Buffer of length N will store N-1 plus the last received message since last time it was cleared.

A FIFO Buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in Figure 20-11.

NOTE: All message objects of a FIFO buffer needs to be read and cleared before the next batch of messages can be stored. Otherwise true FIFO functionality can not be guaranteed, since the message objects of a partly read buffer will be re-filled according to the normal (descending) priority.

Reading from a FIFO Buffer message object and resetting its NewDat bit is handled the same way as reading from a single message object.

Figure 20-11. CPU Handling of a FIFO Buffer (Interrupt Driven)



20.2.9 CAN Message Transfer

Once the DCAN is initialized and the Init bit is reset to 0, the CAN Core synchronizes itself to the CAN bus and is ready for message transfer as per the configured message objects.

The CPU may enable the interrupt lines (setting IE0 and IE1 to 1) at the same time when it clears Init and CCE. The status interrupts EIE and SIE may be enabled simultaneously.

The CAN communication can be carried out in any of the following two modes:

1. Interrupt mode
2. Polling mode.

The Interrupt Register points to those message objects with IntPnd = 1. It is updated even if the interrupt lines to the CPU are disabled (IE0 and IE1 are 0).

The CPU may poll all Message Object's NewDat and TxRqst bits in parallel from the NewData X Registers and the Transmission Request X Registers. Polling can be made easier if all Transmit Objects are grouped at the low numbers, all Receive Objects are grouped at the high numbers.

Received messages are stored into their appropriate message objects if they pass acceptance filtering.

The whole message (including all arbitration bits, DLC and up to eight data bytes) is stored into the message object. As a consequence, when the identifier mask is used, the arbitration bits that are masked to "don't care" may change in the message object when a received message is stored.

The CPU may read or write each message at any time via the Interface Registers, as the Message Handler guarantees data consistency in case of concurrent accesses (for reconfiguration, see Section 20.2.7.6)

If a permanent message object (arbitration and control bits set up during configuration and leaving unchanged for multiple CAN transfers) exists for the message, it is possible to only update the data bytes.

If several transmit messages should be assigned to one message object, the whole message object has to be configured before the transmission of this message is requested.

The transmission of multiple message objects may be requested at the same time. They are subsequently transmitted, according to their internal priority.

Messages may be updated or set to not valid at any time, even if a requested transmission is still pending (for reconfiguration, see Section 20.2.7.7). However, the data bytes will be discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission may be automatically requested by the reception of a remote frame with a matching identifier.

20.2.9.1 Automatic Retransmission

According to the CAN Specification (ISO11898), the DCAN provides a mechanism to automatically retransmit frames that have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed.

By default, this automatic retransmission is enabled. It can be disabled by setting bit DAR (Disable Automatic Retransmission) in CAN Control Register. Further details to this mode are provided in Section 20.2.8.3.

20.2.9.2 Auto-Bus-On

Per default, after the DCAN has entered Bus-Off state, the CPU can start a Bus-Off-Recovery sequence by resetting Init bit. If this is not done, the module will stay in Bus-Off state.

The DCAN provides an automatic Auto-Bus-On feature that is enabled by bit ABO in CAN Control Register. If set, the DCAN will automatically start the Bus-Off-Recovery sequence. The sequence can be delayed by a user-defined number of VCLK cycles that can be defined in Auto-Bus-On Time Register.

NOTE: If the DCAN goes Bus-Off due to massive occurrence of CAN bus errors, it stops all bus activities and automatically sets the Init bit. Once the Init bit has been reset by the CPU or due to the Auto-Bus-On feature, the device will wait for 129 occurrences of Bus Idle (equal to 129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset.

20.2.10 Interrupt Functionality

Interrupts can be generated on two interrupt lines:

1. DCAN0INT line
2. DCAN1INT line

These lines can be enabled by setting the IE0 and IE1 bits, respectively, in the CAN Control Register.

The DCAN provides three groups of interrupt sources: Message Object Interrupts, Status Change Interrupts and Error Interrupts (see Figure 20-12 and Figure 20-13).

The source of an interrupt can be determined by the interrupt identifiers Int0ID / Int1ID in the Interrupt Register (see Section 20.3.5). When no interrupt is pending, the register will hold the value zero.

Each interrupt line remains active until the dedicated field in the Interrupt Register DCAN INT (Int0ID / Int1ID) again reach zero (this means the cause of the interrupt is reset), or until IE0 / IE1 are reset.

The value 0x8000 in the Int0ID field indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Error and Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits WakeUpPnd, RxOk, TxOk and LEC by reading the Error and Status Register DCAN ES, but a write access of the CPU will never generate or reset an interrupt.

Values between 1 and the number of the last message object indicates that the source of the interrupt is one of the message objects, Int0ID resp. Int1ID will point to the pending message interrupt with the highest priority. The Message Object 1 has the highest priority, the last message object has the lowest priority.

An interrupt service routine that reads the message that is the source of the interrupt, may read the message and reset the message object's IntPnd at the same time (ClrIntPnd bit in the IF1/IF2 Command Register). When IntPnd is cleared, the Interrupt Register will point to the next message object with a pending interrupt.

20.2.10.1 Message Object Interrupts

Message Object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxIE and RxIE, that are described in Section 20.2.5.1.

Message Object interrupts can be routed to either DCAN0INT or DCAN1INT line, controlled by the Interrupt Multiplexer Register (see Section 20.3.18).

20.2.10.2 Status Change Interrupts

The events WakeUpPnd, RxOk, TxOk and LEC in Error and Status Register (DCAN ES) belong to the Status Change Interrupts. The Status Change Interrupt group can be enabled by bit in CAN Control Register.

If SIE is set, a Status Change Interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the Message RAM configuration.

Status Change interrupts can only be routed to interrupt line DCAN0INT that has to be enabled by setting IE0 in the CAN Control Register.

NOTE: Reading the Error and Status Register will clear the WakeUpPnd flag. If in global power-down mode, the WakeUpPnd flag is cleared by such a read access before the DCAN module has been waken up by the system, the DCAN may re-assert the WakeUpPnd flag, and a second interrupt may occur (additional information can be found in Section 20.2.11.2).

20.2.10.3 Error Interrupts

The events PER, BOff and EWarn (monitored in Error and Status Register, DCAN ES) belong to the Error Interrupts. The Error Interrupt group can be enabled by setting bit EIE in CAN Control Register.

Error interrupts can only be routed to interrupt line DCAN0INT that has to be enabled by setting IE0 in the CAN Control Register.

Figure 20-12. CAN Interrupt Topology 1

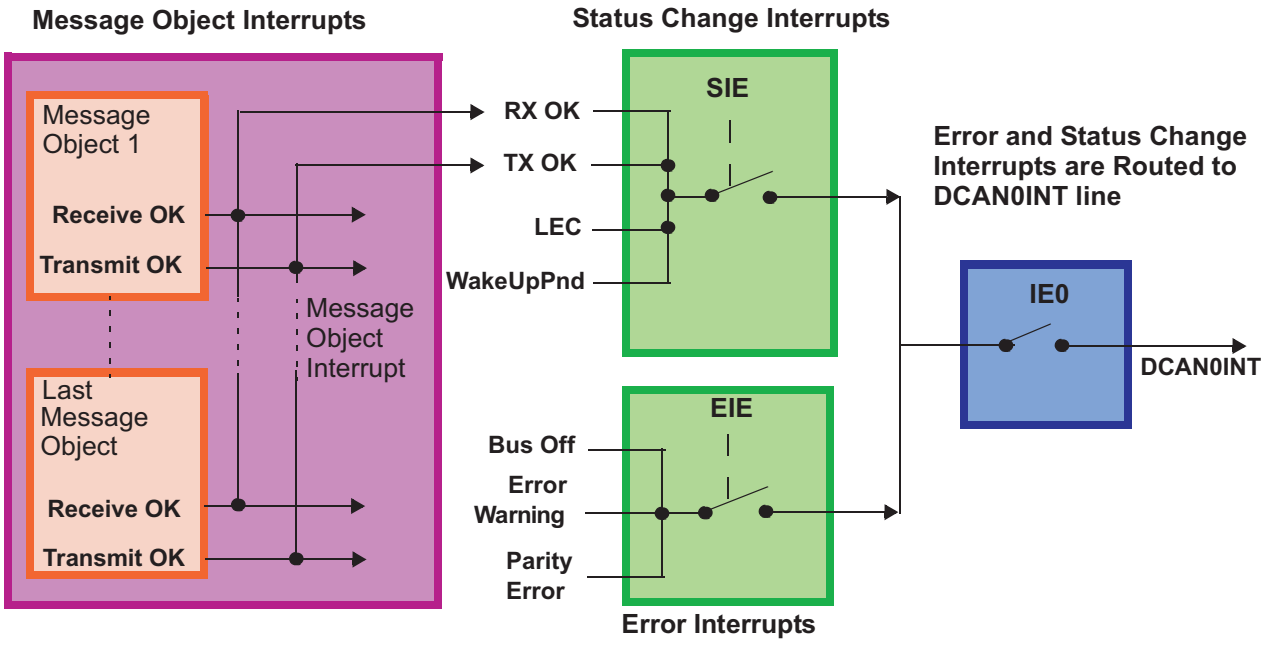
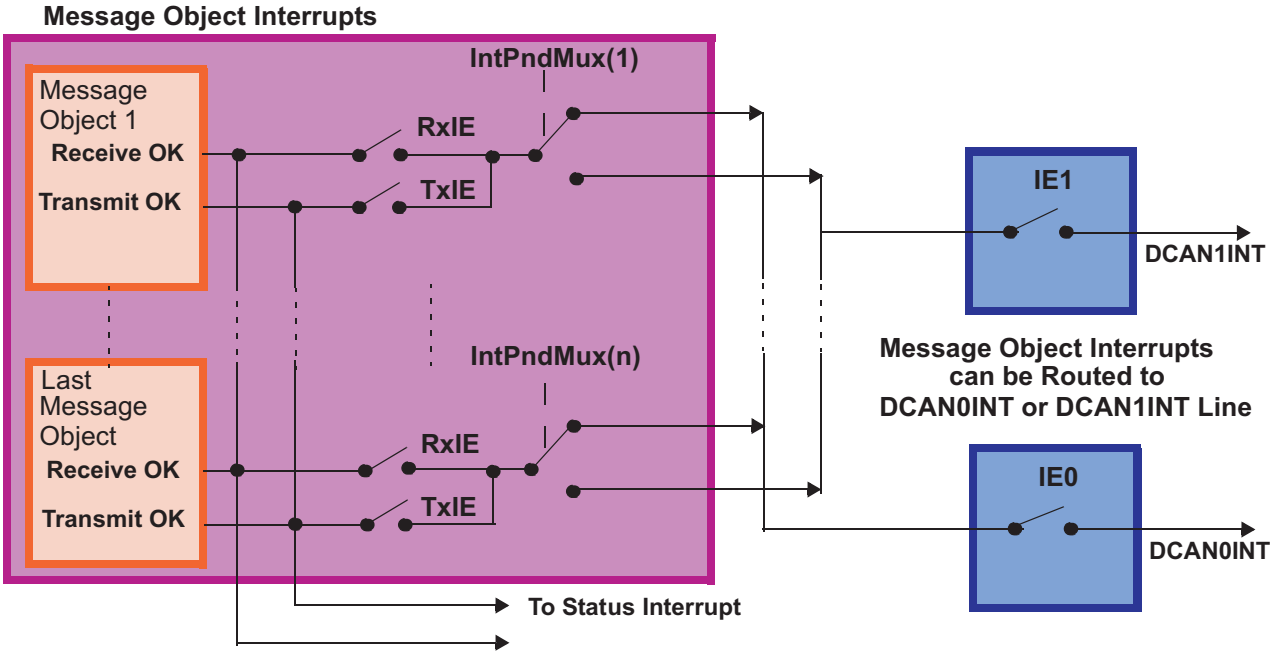


Figure 20-13. CAN Interrupt Topology 2



20.2.11 Global Power-Down Mode

The device architecture supports a centralized global power down control over the peripheral modules through the Peripheral Central Resource (PCR) module (Additional information can be found in Platform Architecture Specification).

20.2.11.1 Entering Global Power-Down Mode

The global power-down mode for the DCAN is requested by setting the appropriate peripheral power down set bit (PSPWRDWNSETx) in the PCR module.

The DCAN then finishes all transmit requests of the message objects. When all requests are done, the DCAN waits until a bus idle state is recognized. Then it will automatically set the Init bit to indicate that the global power-down mode has been entered.

20.2.11.2 Wakeup From Global Power-Down Mode

When the DCAN module is in global power-down mode, a CAN bus activity detection circuit exists, which can be active, if enabled. If this circuit is active,on occurrence of a dominant CAN bus level, the DCAN will set the WakeUpPnd bit in Error and Status Register (DCAN ES).

If Status Interrupts are enabled, also an interrupt will be generated. This interrupt could be used by the application to wakeup the DCAN. For this, the application needs to set the appropriate peripheral power down clear bit (PSPWRDWNCLRx) in the PCR module, and to clear the Init bit in CAN Control Register.

After the Init bit has been cleared, the DCAN module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes Bus-Active again.

NOTE: The CAN transceiver circuit has to stay active during CAN bus activity detection. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power-down mode is lost.

20.2.12 Local Power-Down Mode

Besides the centralized power down mechanism controlled by the PCR module (global power down, see Section 20.2.15), the DCAN supports a local power-down mode that can be controlled within the DCAN control registers.

20.2.12.1 Entering Local Power-Down Mode

The local power-down mode is requested by setting the PDR bit in CAN Control Register.

The DCAN then finishes all transmit requests of the message objects. When all requests are done, DCAN waits until a bus idle state is recognized. Then it will automatically set the Init bit in CAN Control Register to prevent any further CAN transfers, and it will also set the PDA bit in CAN Error and Status Register. With setting the PDA bits, the DCAN module indicates that the local power-down mode has been entered.

During local power-down mode, the internal clocks of the DCAN module are turned off, but there is a wake-up logic (see Section 20.2.12.2) that can be active, if enabled. Also the actual contents of the control registers can be read back.

NOTE: In local low power mode, the application should not clear the Init bit while PDR is set. If there are any messages in the Message RAM configured as to be transmitted and the application resets the Init bit, these messages may be sent.

20.2.12.2 Wakeup From Local Power-Down Mode

There are two ways to wake up the DCAN from local power-down mode:

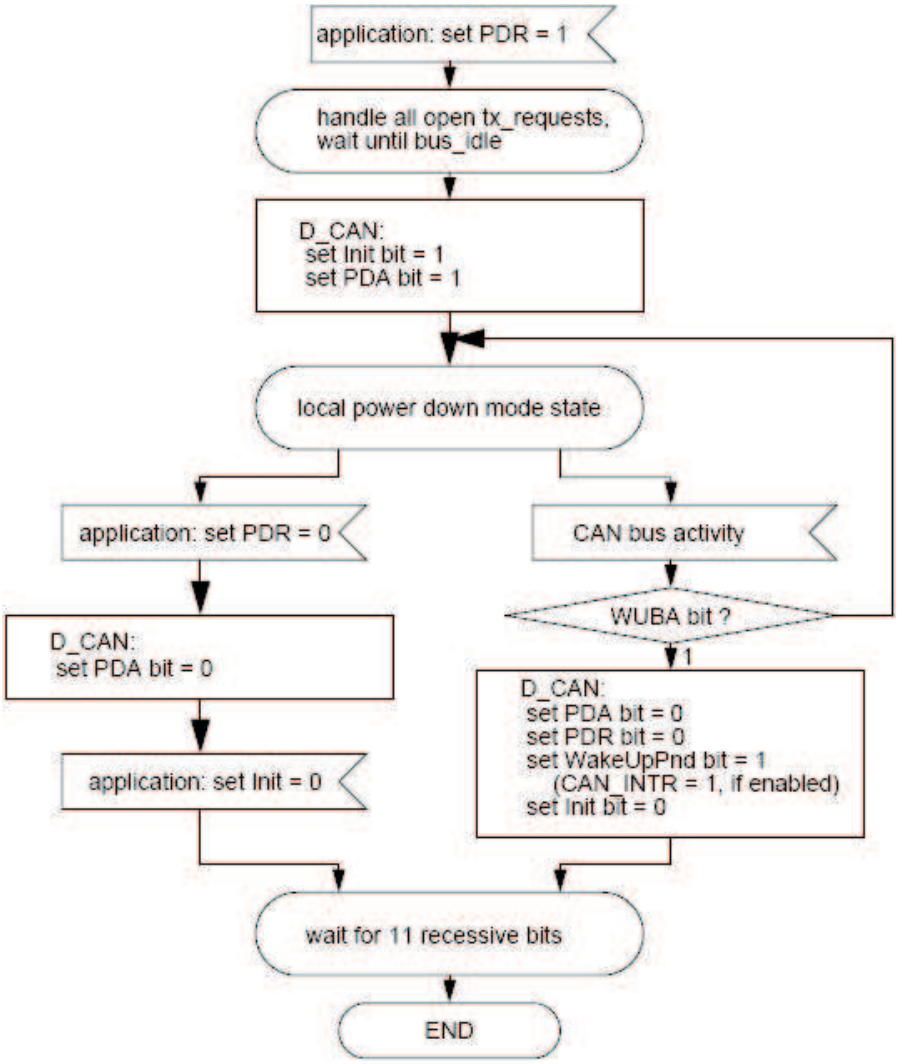
1. The application could wake up the DCAN module manually by clearing the PDR bit and then clearing the Init bit in CAN Control Register.
2. Alternatively, a CAN bus activity detection circuit can be activated by setting the wake up on bus activity bit (WUBA) in CAN Control Register. If this circuit is active, on occurrence of a dominant CAN bus level, the DCAN will automatically start the wake up sequence. It will clear the PDR bit in CAN Control Register and also clear the PDA bit in Error and Status Register. The WakeUpPnd bit in CAN Error and Status Register will be set. If Status Interrupts are enabled, also an interrupt will be generated. Finally the Init bit in CAN control register will be cleared.

After the Init bit has been cleared, the module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes Bus-Active again.

NOTE: The CAN transceiver circuit has to stay active while CAN bus observation. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power-down and automatic wake-up mode, is lost.

Figure 20-14 shows a flow diagram about entering and leaving local power-down mode.

Figure 20-14. Local Power-Down Mode Flow Diagram



20.2.13 GIO Support

The CAN_RX and CAN_TX pins of each DCAN module can be used as general purpose IO pins, if CAN functionality is not needed. This function is controlled by the CAN TX IO Control register (see Section 20.3.30) and the CAN RX IO Control register (see Section 20.3.31).

20.2.14 Test Modes

The DCAN provides several test modes that are mainly intended for production tests or self test.

For all test modes, the Test bit in the CAN Control Register needs to be set to 1. This enables write access to the Test Register.

NOTE: When using any of the Loop Back modes, it must be ensured by software that all message transfers are finished before setting the Init bit to 1.

20.2.14.1 Silent Mode

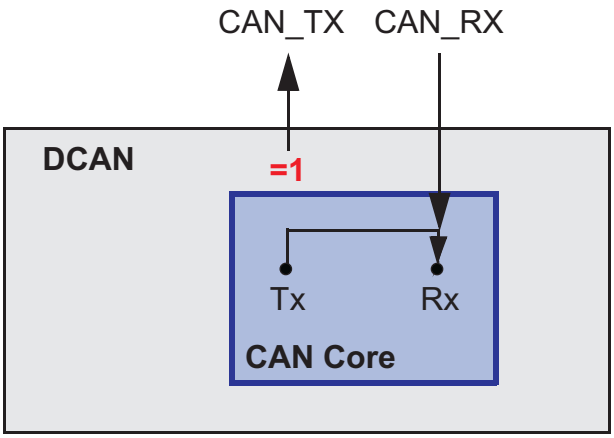
The Silent Mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (for example, acknowledge bit, overload flag, active error flag). The DCAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, these are internally routed to the CAN Core.

Figure 20-15 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.

Silent Mode can be activated by setting the Silent bit in Test Register to 1.

In ISO 11898-1, the Silent Mode is called the Bus Monitoring Mode.

Figure 20-15. CAN Core in Silent Mode



20.2.14.2 Loop Back Mode

The Loop Back Mode is mainly intended for hardware self-test functions. In this mode, the CAN Core uses internal feedback from Tx output to Rx input. Transmitted messages are treated as received messages, and can be stored into message objects if they pass acceptance filtering. The actual value of the CAN_RX input pin is disregarded by the CAN Core. Transmitted messages still can be monitored at the CAN_TX pin.

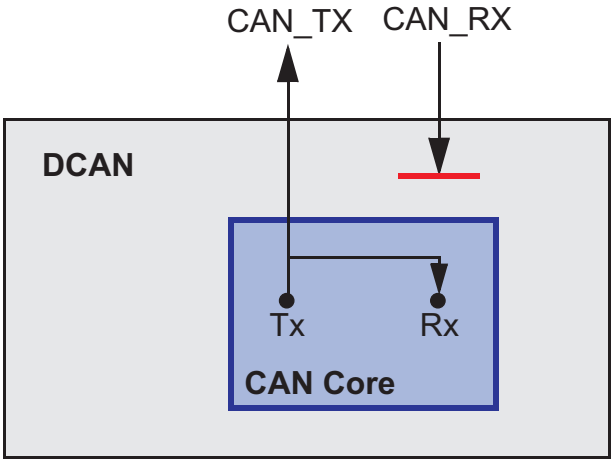
In order to be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode.

Figure 20-16 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Loop Back Mode.

Loop Back Mode can be activated by setting the LBack bit in Test Register to 1.

NOTE: In Loop Back mode, the signal path from CAN Core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN Core are disregarded. For including these into the testing, see External Loop Back mode (Section 20.2.14.3).

Figure 20-16. CAN Core in Loop Back Mode



20.2.14.3 External Loop Back Mode

The External Loop Back Mode is similar to the Loop Back Mode, however it includes the signal path from CAN Core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN Core. When External Loop Back Mode is selected, the input of the CAN core is connected to the input buffer of the Tx pin.

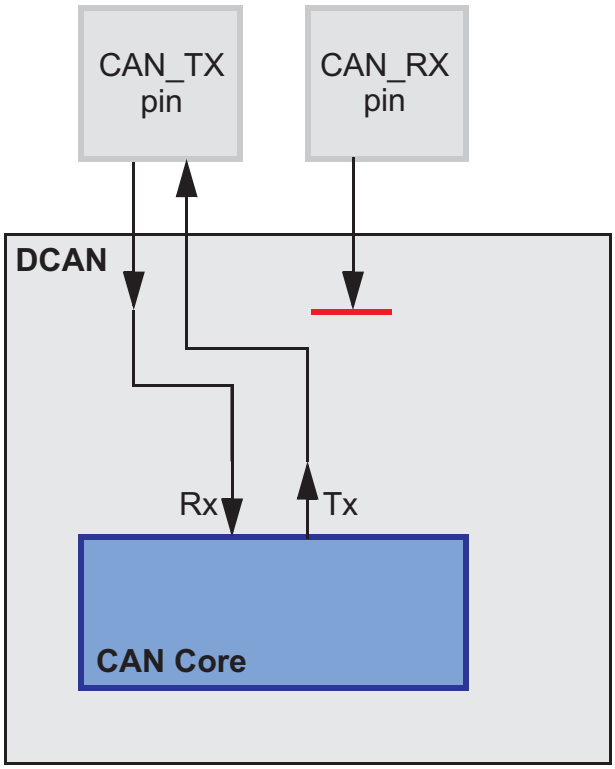
With this configuration, the Tx pin IO circuit can be tested.

External Loop Back Mode can be activated by setting the EXL bit in Test Register to 1.

Figure 20-17 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in External Loop Back Mode.

NOTE: When Loop Back Mode is active (LBack bit is set), the EXL bit will be ignored.

Figure 20-17. CAN Core in External Loop Back Mode

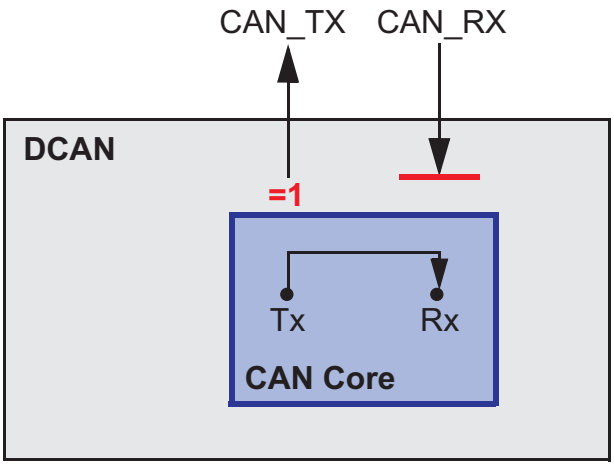


20.2.14.4 Loop Back Combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by setting the LBack and Silent bits at the same time. This mode can be used for a “Hot Self-test”, that is, the DCAN hardware can be tested without affecting the CAN network. In this mode, the CAN_RX pin is disconnected from the CAN Core and no dominant bits will be sent on the CAN_TX pin.

Figure 20-18 shows the connection of the signals CAN_TX and CAN_RX to the CAN Core in case of the combination of Loop Back Mode with Silent Mode.

Figure 20-18. CAN Core in Loop Back Combined with Silent Mode



20.2.14.5 Software Control of CAN_TX Pin

Four output functions are available for the CAN transmit pin CAN_TX. Additionally to its default function (serial data output), the CAN_TX pin can drive constant dominant or recessive values, or it can drive the CAN Sample Point signal to monitor the CAN Core’s bit timing.

Combined with the readable value of the CAN_RX pin, this can be used to check the physical layer of the CAN bus.

The output mode of pin CAN_TX is selected by programming the Test Register Tx bits as described in Section 20.3.6.

NOTE: The software control for pin CAN_TX interferes with CAN protocol functions. For CAN message transfer or any of the test modes, Loop Back Mode, External Loop Back Mode, or Silent Mode, the CAN_TX pin should operate in its default functionality.

20.2.15 Parity Check Mechanism

The DCAN provides a parity check mechanism to ensure data integrity of Message RAM data. For each word (32 bits) in Message RAM, one parity bit will be calculated. The formation of the different words is according to the Message RAM representation in RDA mode (see Section 20.2.5.4).

Parity information is stored in the Message RAM on write accesses and will be checked against the stored parity bit from Message RAM on read accesses.

The parity check functionality can be enabled or disabled by PMD bit field in CAN Control Register.

In case of disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the DCAN. The parity bits could be read in Debug/Suspend mode (see Section 20.2.5.3) or in RDA mode (see Section 20.2.5.4). However, direct write access to the parity bits is only possible in this two modes with parity check disabled.

A parity bit will be set, if the modulo-2-sum of the data bits is 1. This definition is equivalent to: The parity bit will be set, if the number of 1 bits in the data is odd.

NOTE: Parity scheme can be changed via the System module DEV Parity Control Register1 on device basis for all peri RAMs.

20.2.15.1 Behavior on Parity Error

On any read access to Message RAM (for example, during start of a CAN frame transmission), the parity of the message object will be checked. If a parity error is detected, the PER bit in Error and Status Register will be set. If error interrupts are enabled, also an interrupt would be generated. In order to avoid the transmission of invalid data over the CAN bus, the d bit of the message object will be reset.

The message object data can be read by the host CPU, independently of parity errors. Thus, the application has to ensure that the read data is valid, for example, by immediately checking the Parity Error Code register on parity error interrupt.

NOTE: During RAM initialization, no parity check will be done.

20.2.15.2 Parity Testing

Testing the parity mechanism can be done by enabling the bit RDA (RamDirectAccess) and manually writing the parity bits directly to the dedicated RAM locations. With this, data and parity bits could be checked when reading directly from RAM.

NOTE: If parity check was disabled, the application has to ensure correct parity bit handling in order to prevent parity errors later on when parity check is enabled.

20.2.16 Debug/Suspend Mode

When the CPU is halted during debug, all DCAN registers are visible and can be inspected and modified by the CPU.

In addition, the Message RAM is directly memory-mapped as described in Table 20-3.

The CAN controller provides two options for entering the debug/suspend state. The options are controlled by the IDS bit in the CAN Control Register (DCAN CTL). By default, when IDS is 0, the DCAN controller completes any active transfers on the CAN bus and waits until the bus is idle before halting. When IDS is 1, the DCAN halts immediately as soon as the CPU is halted.

The InitDbg bit in DCAN CTL register indicates when the DCAN controller has actually entered the debug/suspend state.

NOTE: During Debug/Suspend Mode, the Message RAM cannot be accessed via the IFx register sets.

Writing to control registers in debug/suspend mode may influence the CAN state machine and further message handling.

For debug support, the auto clear functionality of the following DCAN registers is disabled:

- Error and Status Register (clear of status flags by read)
- IF1/IF2 Command Registers

20.3 DCAN Control Registers

Table 20-6 lists the control registers of the DCAN. After hardware reset, the registers of the DCAN hold the values shown in the register descriptions. The base address for the control registers is FFF7 DC00h for DCAN1 and FFF7 DE00h for DCAN2.

Additionally, the Bus-Off state is reset and the CAN_TX pin is set to recessive (HIGH). The Init bit in the CAN Control Register is set to enable the software initialization. The DCAN will not influence the CAN bus until the CPU resets Init to 0.

Table 20-6. DCAN Control Registers

Offset	Acronym	Register Description	Section
00h	DCAN CTL	CAN Control Register	Section 20.3.1
04h	DCAN ES	Error and Status Register	Section 20.3.2
08h	DCAN ERRC	Error Counter Register	Section 20.3.3
0Ch	DCAN BTR	Bit Timing Register	Section 20.3.4
10h	DCAN INT	Interrupt Register	Section 20.3.5
14h	DCAN TEST	Test Register	Section 20.3.6
1Ch	DCAN PERR	Parity Error Code Register	Section 20.3.7
20h	DCAN REL	Core Release Register	Section 20.3.8
80h	DCAN ABOTR	Auto-Bus-On Time Register	Section 20.3.9
84h	DCAN TXRQX	Transmission Request X Register	Section 20.3.10
88h	DCAN TXRQ12	Transmission Request 12 Register	Section 20.3.11
8Ch	DCAN TXRQ34	Transmission Request 34 Register	Section 20.3.11
90h	DCAN TXRQ56	Transmission Request 56 Register	Section 20.3.11
94h	DCAN TXRQ78	Transmission Request 78 Register	Section 20.3.11
98h	DCAN NWDATX	New Data X Register	Section 20.3.12
9Ch	DCAN NWDAT12	New Data 12 Register	Section 20.3.13
A0h	DCAN NWDAT34	New Data 34 Register	Section 20.3.13
A4h	DCAN NWDAT56	New Data 56 Register	Section 20.3.13
A8h	DCAN NWDAT78	New Data 78 Register	Section 20.3.13
ACh	DCAN INTPNDX	Interrupt Pending X Register	Section 20.3.14

Table 20-6. DCAN Control Registers (continued)

Offset	Acronym	Register Description	Section
B0h	DCAN INTPND12	Interrupt Pending 12 Register	Section 20.3.15
B4h	DCAN INTPND34	Interrupt Pending 34 Register	Section 20.3.15
B8h	DCAN INTPND56	Interrupt Pending 56 Register	Section 20.3.15
BCh	DCAN INTPND78	Interrupt Pending 78 Register	Section 20.3.15
C0h	DCAN MSGVALX	Message Valid X Register	Section 20.3.16
C4h	DCAN MSGVAL12	Message Valid 12 Register	Section 20.3.17
C8h	DCAN MSGVAL34	Message Valid 34 Register	Section 20.3.17
CCh	DCAN MSGVAL56	Message Valid 56 Register	Section 20.3.17
D0h	DCAN MSGVAL78	Message Valid 78 Register	Section 20.3.17
D8h	DCAN INTMUX12	Interrupt Multiplexer 12 Register	Section 20.3.18
DCh	DCAN INTMUX34	Interrupt Multiplexer 34 Register	Section 20.3.18
E0h	DCAN INTMUX56	Interrupt Multiplexer 56 Register	Section 20.3.18
E4h	DCAN INTMUX78	Interrupt Multiplexer 78 Register	Section 20.3.18
100h	DCAN IF1CMD	IF1 Command Register	Section 20.3.19
104h	DCAN IF1MSK	IF1 Mask Register	Section 20.3.20
108h	DCAN IF1ARB	IF1 Arbitration Register	Section 20.3.21
10Ch	DCAN IF1MCTL	IF1 Message Control Register	Section 20.3.22
110h	DCAN IF1DATA	IF1 Data A Register	Section 20.3.23
114h	DCAN IF1DATB	IF1 Data B Register	Section 20.3.23
120h	DCAN IF2CMD	IF2 Command Register	Section 20.3.19
124h	DCAN IF2MSK	IF2 Mask Register	Section 20.3.20
128h	DCAN IF2ARB	IF2 Arbitration Register	Section 20.3.21
12Ch	DCAN IF2MCTL	IF2 Message Control Register	Section 20.3.22
130h	DCAN IF2DATA	IF2 Data A Register	Section 20.3.23
134h	DCAN IF2DATB	IF2 Data B Register	Section 20.3.23
140h	DCAN IF3OBS	IF3 Observation Register	Section 20.3.24
144h	DCAN IF3MSK	IF3 Mask Register	Section 20.3.25
148h	DCAN IF3ARB	IF3 Arbitration Register	Section 20.3.26
14Ch	DCAN IF3MCTL	IF3 Message Control Register	Section 20.3.27
150h	DCAN IF3DATA	IF3 Data A Register	Section 20.3.28
154h	DCAN IF3DATB	IF3 Data B Register	Section 20.3.28
160h	DCAN IF3UPD12	IF3 Update Enable 12 Register	Section 20.3.29
164h	DCAN IF3UPD34	IF3 Update Enable 34 Register	Section 20.3.29
168h	DCAN IF3UPD56	IF3 Update Enable 56 Register	Section 20.3.29
16Ch	DCAN IF3UPD78	IF3 Update Enable 78 Register	Section 20.3.29
1E0h	DCAN TIOC	CAN TX IO Control Register	Section 20.3.30
1E4h	DCAN RIOC	CAN RX IO Control Register	Section 20.3.31

20.3.1 CAN Control Register (DCAN CTL)

Figure 20-19. CAN Control Register (DCAN CTL) [offset = 00]

31						26	25	24
Reserved							WUBA	PDR
R-0							R/W-0	R/W-0
23	22	21	20	19	18		17	16
Reserved							IE1	InitDbg
R-0	R-0	R-0	R/W-0	R/W-0	R/W-0		R/W-0	R-0
15	14	13				10	9	8
SWR	Reserved	PMD					ABO	IDS
R/WP-0	R-0	R/W-5h					R/W-0	R/W-0
7	6	5	4	3	2		1	0
Test	CCE	DAR	Reserved	EIE	SIE		IE0	Init
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0		R/W-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; WP = Write protected by Init bit; -*n* = value after reset

Table 20-7. CAN Control Register Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	These bits are always read as 0. Writes have no effect.
25	WUBA		Automatic wake up on bus activity when in local power-down mode.
		0	No detection of a dominant CAN bus level while in local power-down mode.
		1	Detection of a dominant CAN bus level while in local power-down mode is enabled. On occurrence of a dominant CAN bus level, the wake up sequence is started. (Additional information can be found in Section 20.2.12.)
			Note: The CAN message, which Initiates the bus activity, cannot be received. This means that the first message received in power-down and automatic wake-up mode, will be lost.
24	PDR		Request for local low power-down mode.
		0	No application request for local low power-down mode. If the application has cleared this bit while DCAN in local power-down mode, also the Init bit has to be cleared.
		1	Local power-down mode has been requested by application. The DCAN will acknowledge the local power-down mode by setting bit PDA in Error and Status Register. The local clocks will be turned off by DCAN internal logic (Additional information can be found in Section 20.2.12).
23-21	Reserved	0	These bits are always read as 0. Writes have no effect.
20-18	Reserved	0	Reserved. Do not use.
17	IE1		Interrupt line 1 enable.
		0	Disabled. Module Interrupt DCAN1INT is always low.
		1	Enabled. Interrupts will assert line DCAN1INT to 1; line remains active until pending interrupts are processed.
16	InitDbg		Internal Init state while debug access.
		0	Not in debug mode, or debug mode requested but is not entered.
		1	Debug mode requested and is internally entered; the DCAN is ready for debug accesses.
15	SWR		SW Reset enable.
		0	Normal operation.
		1	Module is forced to reset state. This bit will automatically get cleared after execution of SW reset after one VBUSP clock cycle.
			Note: To execute SW reset the following procedure is necessary: 1. Set Init bit to shut down CAN communication. 2. Set SWR bit additionally to Init bit.
14	Reserved	0	This bit is always read as 0. Writes have no effect.

Table 20-7. CAN Control Register Field Descriptions (continued)

Bit	Field	Value	Description
13-10	PMD		Parity enable.
		5h	Parity function is disabled.
		Others	Parity function is enabled.
9	ABO		Auto-Bus-On enable.
		0	The Auto-Bus-On feature is disabled.
		1	The Auto-Bus-On feature is enabled.
8	IDS		Interruption Debug Support enable.
		0	When Debug/Suspend mode is requested, DCAN will wait for a started transmission or reception to be completed before entering Debug/Suspend mode.
		1	When Debug/Suspend mode is requested, DCAN will interrupt any transmission or reception, and enter Debug/Suspend mode immediately.
7	Test		Test Mode enable.
		0	Normal operation
		1	Test mode
6	CCE		Configuration Change Enable.
		0	The CPU has no write access to the BTR config register.
		1	The CPU has write access to the BTR config register (when Init bit is set).
5	DAR		Disable Automatic Retransmission.
		0	Automatic Retransmission of not successful messages is enabled.
		1	Automatic Retransmission is disabled.
4	Reserved	0	This bit is always read as 0. Writes have no effect.
3	EIE		Error Interrupt Enable.
		0	Disabled. PER, BOff, and EWarn bits cannot generate an interrupt.
		1	Enabled. PER, BOff, and EWarn bits can generate an interrupt at DCAN0INT line and affect the Interrupt Register.
2	SIE		Status Change Interrupt Enable.
		0	Disabled. WakeUpPnd, RxOk, TxOk, and LEC bits cannot generate an interrupt.
		1	Enabled. WakeUpPnd, RxOk, TxOk, and LEC can generate an interrupt at DCAN0INT line and affect the Interrupt Register.
1	IE0		Interrupt line 0 enable.
		0	Disabled. Module Interrupt DCAN0INT is always low.
		1	Enabled. Interrupts will assert line DCAN0INT to 1; line remains active until pending interrupts are processed.
0	Init		Initialization.
		0	Normal operation.
		1	Initialization mode is entered.

NOTE: The Bus-Off recovery sequence (see CAN specification) cannot be shortened by setting or resetting Init bit. If the module goes Bus-Off, it will automatically set the Init bit and stop all bus activities.

When the Init bit is cleared by the application again, the module will then wait for 129 occurrences of Bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset.

After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 error code is written to the Error and Status Register, enabling the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the Bus-Off recovery sequence.

20.3.2 Error and Status Register (DCAN ES)

Figure 20-20. Error and Status Register (DCAN ES) [offset = 04h]

31															16
Reserved															
R-0															
15										11	10	9			8
Reserved											PDA	WakeUpPnd			PER
R-0											R-0	R/C-0			R/C-0
7		6		5		4		3		2					0
BOff		EWarn		EPass		RxOK		TxOK		LEC					
R-0		R-0		R-0		R/C-0		R/C-0		R/S-7h					

LEGEND: R = Read only; C = Clear; S = Set; -*n* = value after reset

Table 20-8. Error and Status Register Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	These bits are always read as 0. Writes have no effect.
10	PDA		Local power-down mode acknowledge.
		0	DCAN is not in local power-down mode.
		1	Application request for setting DCAN to local power-down mode was successful. DCAN is in local power-down mode.
9	WakeUpPnd		Wake Up Pending
			This bit can be used by the CPU to identify the DCAN as the source to wake up the system.
		0	No Wake Up is requested by DCAN.
		1	DCAN has initiated a wake up of the system due to dominant CAN bus while module power down. This bit will be reset if Error and Status Register is read.
8	PER		Parity Error Detected
		0	No parity error has been detected since last read access.
		1	The parity check mechanism has detected a parity error in the Message RAM. This bit will be reset if Error and Status Register is read.
7	BOff		Bus-Off State
		0	The CAN module is not Bus-Off state.
		1	The CAN module is in Bus-Off state.
6	EWarn		Warning State
		0	Both error counters are below the error warning limit of 96.
		1	At least one of the error counters has reached the error warning limit of 96.
5	EPass		Error Passive State
		0	On CAN Bus error, the DCAN could send active error frames.
		1	The CAN Core is in the error passive state as defined in the CAN Specification.
4	RxOK		Received a message successfully.
		0	No message has been successfully received since the last time when this bit was read by the CPU. This bit is never reset by DCAN internal events.
		1	A message has been successfully received since the last time when this bit was reset by a read access of the CPU (independent of the result of acceptance filtering).This bit will be reset if Error and Status Register is read.
3	TxOK		Transmitted a message successfully.
		0	No message has been successfully transmitted since the last time when this bit was read by the CPU. This bit is never reset by DCAN internal events.
		1	A message has been successfully transmitted (error free and acknowledged by at least one other node) since the last time when this bit was reset by a read access of the CPU. This bit will be reset if Error and Status Register is read.

Table 20-8. Error and Status Register Field Descriptions (continued)

Bit	Field	Value	Description
2-0	LEC		Last Error Code
			The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error.
		0	No Error
		1h	Stuff Error: More than five equal bits in a row have been detected in a part of a received message where this is not allowed.
		2h	Form Error: A fixed format part of a received frame has the wrong format.
		3h	Ack Error: The message this CAN Core transmitted was not acknowledged by another node.
		4h	Bit1 Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
		5h	Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value 0), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
		6h	CRC Error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).
		7h	No CAN bus event was detected since the last time when CPU has read the Error and Status Register. Any read access to the Error and Status Register re-Initializes the LEC to value 7h.

Interrupts are generated by bits PER, BOff, and EWarn (if EIE bit in CAN Control Register is set) and by bits WakeUpPnd, RxOk, TxOk, and LEC (if SIE bit in CAN Control Register is set). A change of bit EPass will not generate an Interrupt.

NOTE: Reading the Error and Status Register clears the WakeUpPnd, PER, RxOk, and TxOk bits and sets the LEC to value 7h. Additionally, the Status Interrupt value (8000h) in the Interrupt Register will be replaced by the next lower priority interrupt value.

For debug support, the auto clear functionality of Error and Status Register (clear of status flags by read) is disabled when in Debug/Suspend mode.

20.3.3 Error Counter Register (DCAN ERRC)

Figure 20-21. Error Counter Register (DCAN ERRC) [offset = 08h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC							TEC							
R-0	R-0							R-0							

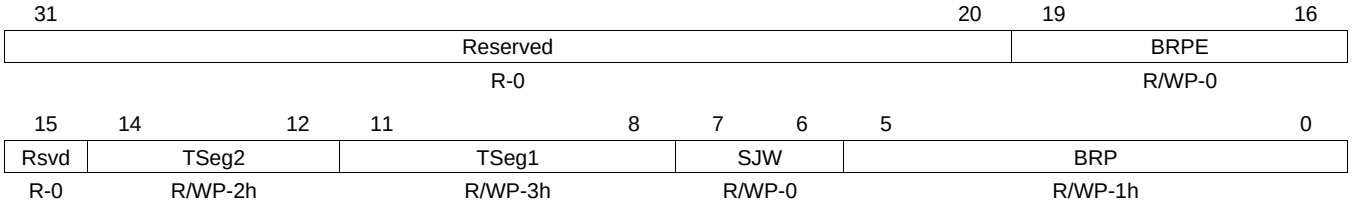
LEGEND: R = Read only; -*n* = value after reset

Table 20-9. Error Counter Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These bits are always read as 0. Writes have no effect.
15	RP		Receive Error Passive
		0	The Receive Error Counter is below the error passive level.
		1	The Receive Error Counter has reached the error passive level as defined in the CAN Specification.
14-8	REC	0-7Fh	Receive Error Counter. Actual state of the Receive Error Counter.
7-0	TEC	0-FFh	Transmit Error Counter. Actual state of the Transmit Error Counter.

20.3.4 Bit Timing Register (DCAN BTR)

Figure 20-22. Bit Timing Register (DCAN BTR) [offset = 0Ch]



LEGEND: R/W = Read/Write; R = Read only; WP = Write Protected by CCE bit; -*n* = value after reset

Table 20-10. Bit Timing Register Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	These bits are always read as 0. Writes have no effect.
	BRPE	0-Fh	Baud Rate Prescaler Extension. Valid programmed values are 0 to 15. By programming BRPE the Baud Rate Prescaler can be extended to values up to 1024.
15	Reserved	0	This bit is always read as 0. Writes have no effect.
14-12	TSeg2	0-7h	Time segment after the sample point. Valid programmed values are 0 to 7. The actual TSeg2 value that is interpreted for the Bit Timing will be the programmed TSeg2 value + 1.
11-8	TSeg1	1h-Fh	Time segment before the sample point. Valid programmed values are 1 to 15. The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1.
7-6	SJW	0-3h	Synchronization Jump Width Valid programmed values are 0 to 3.The actual SJW value interpreted for the Synchronization will be the programmed SJW value + 1.
5-0	BRP	0-3Fh	Baud Rate Prescaler Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid programmed values are 0 to 63.The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1.

NOTE: This register is only writable if CCE and Init bits in the CAN Control Register are set.

The CAN bit time may be programmed in the range of 8 to 25 time quanta.

The CAN time quantum may be programmed in the range of 1 to 1024 CAN_CLK periods.

With a CAN_CLK of 8 MHz and BRPE = 00, the reset value of 2301h configures the DCAN for a bit rate of 500kBit/s.

For details, see Section 20.2.3.2.1.

20.3.5 Interrupt Register (DCAN INT)

Figure 20-23. Interrupt Register (DCAN INT) [offset = 10h]

31	24	23	16
Reserved		Int1ID	
R-0		R-0	

15	0
Int0ID	
R-0	

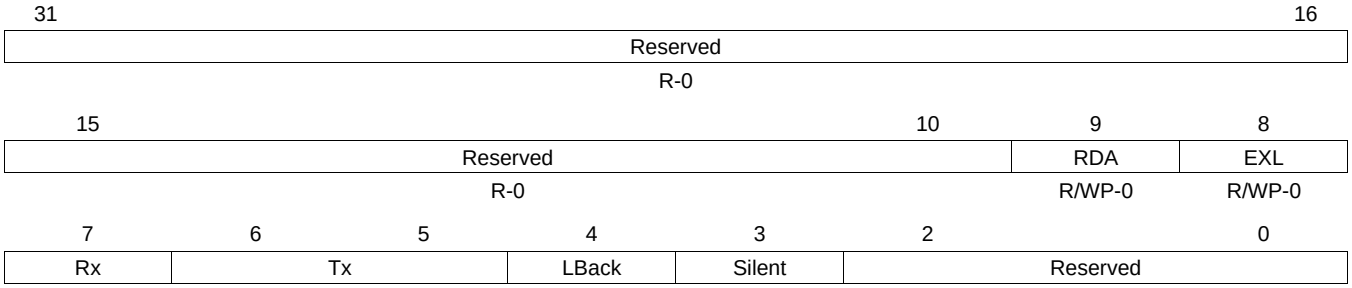
LEGEND: R = Read only; -*n* = value after reset

Table 20-11. Interrupt Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	These bits are always read as 0. Writes have no effect.
23-16	Int1ID		Interrupt 1 Identifier (indicates the message object with the highest-pending interrupt).
		0	No interrupt is pending.
		1h-10h	Number of message object that caused the interrupt for DCAN2.
		1h-20h	Number of message object that caused the interrupt for DCAN1.
		11h-FFh	Reserved for DCAN2.
		21h-FFh	Reserved for DCAN1.
			If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority. The DCAN1INT interrupt line remains active until Int1ID reaches value 0 (the cause of the interrupt is reset) or until IE1 is cleared.
			A message interrupt is cleared by clearing the message object's IntPnd bit.
			Among the message interrupts, the message object's interrupt priority decreases with increasing message number.
15-0	Int0ID		Interrupt Identifier (the number here indicates the source of the interrupt).
		0	No interrupt is pending.
		1h-10h	Number of message object that caused the interrupt for DCAN2.
		1h-20h	Number of message object that caused the interrupt for DCAN1.
		11h-7FFFh	Reserved for DCAN2.
		21h-7FFFh	Reserved for DCAN1.
		8000h	Error and Status Register value is not 0x07.
		8001h-FFFFh	Reserved
			If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority. The DCAN0INT interrupt line remains active until Int0ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared.
			The Status Interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number.

20.3.6 Test Register (DCAN TEST)

Figure 20-24. Test Register (DCAN TEST) [offset = 14h]



LEGEND: R/W = Read/Write; R = Read only; WP = Write Protected by Test bit; -*n* = value after reset; U = Undefined

Table 20-12. Test Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	These bits are always read as 0. Writes have no effect.
9	RDA		RAM Direct Access Enable
		0	Normal operation
		1	Direct access to the RAM is enabled while in Test Mode.
8	EXL		External Loop Back Mode Enable
		0	Disabled
		1	Enabled
7	Rx		Receive Pin. Monitors the actual value of the CAN_RX pin.
		0	The CAN bus is dominant.
		1	The CAN bus is recessive.
6-5	Tx		Control of CAN_TX pin
		0	Normal operation, CAN_TX is controlled by the CAN Core.
		1h	Sample Point can be monitored at CAN_TX pin.
		2h	CAN_TX pin drives a dominant value.
		3h	CAN_TX pin drives a recessive value.
4	LBack		Loop Back Mode Enable
		0	Disabled
		1	Enabled
3	Silent		Silent Mode Enable
		0	Disabled
		1	Enabled
2-0	Reserved	0	These bits are always read as 0. Writes have no effect.

For all test modes, the Test bit in CAN Control Register needs to be set to 1. If Test bit is set, the RDA, EXL, Tx1, Tx0, LBack, and Silent bits are writable. Bit Rx monitors the state of pin CAN_RX and therefore is only readable. All Test Register functions are disabled when the Test bit is cleared to 0.

NOTE: The Test Register is only writable if Test bit in CAN Control Register is set.

Setting Tx[1:0] other than 00 will disturb message transfer.

When the internal loop back mode is active (LBack bit is set), the EXL bit will be ignored.

20.3.7 Parity Error Code Register (DCAN PERR)

Figure 20-25. Parity Error Code Register (DCAN PERR) [offset = 1Ch]

31	16
Reserved	
R-0	

15	14	11	10	8	7	0
Reserved			Word Number		Message Number	
R-0			R-U		R-U	

LEGEND: R = Read only; U = Undefined; -*n* = value after reset

Table 20-13. Parity Error Code Register Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	These bits are always read as 0. Writes have no effect.
10-8	Word Number		Word number where parity error has been detected.
		1h-5h	RDA word number (1 to 5) of the message object (according to the Message RAM representation in RDA mode, see Section 20.2.5.4).
7-0	Message Number	1h-FFh	Message object number where parity error has been detected. For DCAN1, only values 1h-20h are valid; values 21h-FFh are invalid. For DCAN2, only values 1h-10h are valid; values 11h-FFh are invalid.

If a parity error is detected, the PER flag will be set in the Error and Status Register. This bit is not reset by the parity check mechanism; it must be reset by reading the Error and Status Register.

In addition to the PER flag, the Parity Error Code Register will indicate the memory area where the parity error has been detected (message number and word number).

If more than one word with a parity error was detected, the highest word number with a parity error will be displayed.

After a parity error has been detected, the register will hold the last error code until power is removed.

20.3.8 Core Release Register (DCAN REL)

Figure 20-26. Core Release Register (DCAN REL) [offset = 20h]

31	28	27	24	23	20	19	16
REL		STEP		SUBSTEP		YEAR	
R-Ah		R-3h		R-1h		R-7h	

15	8	7	0
MON		DAY	
R-5h		R-4h	

LEGEND: R = Read only; -*n* = value after reset

Table 20-14. Core Release Register (DCAN REL) Field Descriptions

Bit	Field	Value	Description
31-28	REL	0-9h	Core Release. One digit, BCD-coded.
27-24	STEP	0-9h	Step of Core Release. One digit, BCD-coded.
23-20	SUBSTEP	0-9h	Substep of Core Release. One digit, BCD-coded.
19-16	YEAR	0-9h	Design Time Stamp, Year. One digit, BCD-coded. This field is set by constant parameter on DCAN synthesis.
15-8	MON	0-12h	Design Time Stamp, Month. Two digits, BCD-coded. This field is set by constant parameter on DCAN synthesis.
7-0	DAY	0-31h	Design Time Stamp, Day. Two digits, BCD-coded. This field is set by constant parameter on DCAN synthesis.

20.3.9 Auto-Bus-On Time Register (DCAN ABOTR)

Figure 20-27. Auto-Bus-On Time Register (DCAN ABOTR) [offset = 80h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABO_TIME															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABO_TIME															
R/W-0															

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 20-15. Auto-Bus-On Time Register Field Descriptions

Bit	Field	Value	Description
31-0	ABO_TIME	0-FFFF FFFFh	Number of VBUS clock cycles before a Bus-Off recovery sequence is started by clearing the Init bit. This function has to be enabled by setting the ABO bit in the CAN Control Register. The Auto-Bus-On timer is realized by a 32-bit counter that starts to count down to 0 when the module goes Bus-Off. The counter will be reloaded with the preload value of the ABO_TIME register after this phase.

NOTE: On write access to the CAN Control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted.

During Debug/Suspend mode, running Auto-Bus-On timer will be paused.

20.3.10 Transmission Request X Register (DCAN TXRQ X)

With the Transmission Request X Register, the CPU can detect if one or more bits in the different Transmission Request Registers are set. Each register bit represents a group of eight message objects. If at least one of the TxRqst bits of these message objects are set, the corresponding bit in the Transmission Request X Register will be set.

Figure 20-28. Transmission Request X Register (DCAN TXRQ X) [offset = 84h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxRqstReg8		TxRqstReg7		TxRqstReg6		TxRqstReg5		TxRqstReg4		TxRqstReg3		TxRqstReg2		TxRqstReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Example 1

Bit 0 of the Transmission Request X Register represents byte 0 of the Transmission Request 1 Register. If one or more bits in this byte are set, bit 0 of the Transmission Request X Register will be set.

20.3.11 Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78)

These registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1/IF2 Message Interface Registers, or by the Message Handler after reception of a remote frame or after a successful transmission.

Figure 20-29. Transmission Request 12 Register [offset = 88h]

31	0
TxRqst[32:1]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-30. Transmission Request 34 Register [offset = 8Ch]

31	0
TxRqst[64:33]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-31. Transmission Request 56 Register [offset = 90h]

31	0
TxRqst[96:65]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-32. Transmission Request 78 Register [offset = 94h]

31	0
TxRqst[128:97]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 20-16. Transmission Request Registers Field Descriptions

Bit	Name	Value	Description
31-0	TxRqs[128:1]		Transmission Request Bits (for all message objects).
		0	No transmission has been requested for this message object.
		1	The transmission of this message object is requested and is not yet done.

20.3.12 New Data X Register (DCAN NWDAT X)

With the New Data X Register, the CPU can detect if one or more bits in the different New Data Registers are set. Each register bit represents a group of eight message objects. If at least on of the NewDat bits of these message objects are set, the corresponding bit in the New Data X Register will be set.

Figure 20-33. New Data X Register (DCAN NWDAT X) [offset = 98h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDatReg8		NewDatReg7		NewDatReg6		NewDatReg5		NewDatReg4		NewDatReg3		NewDatReg2		NewDatReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Equation 1

Bit 0 of the New Data X Register represents byte 0 of the New Data 1 Register. If one or more bits in this byte are set, bit 0 of the New Data X Register will be set.

20.3.13 New Data Registers (DCAN NWDAT12 to DCAN NWDAT78)

These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after reception of a data frame or after a successful transmission.

Figure 20-34. New Data 12 Register [offset = 9Ch]

31	0
NewDat[32:1]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-35. New Data 34 Register [offset = A0h]

31	0
NewDat[64:33]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-36. New Data 56 Register [offset = A4h]

31	0
NewDat[96:65]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-37. New Data 78 Register [offset = A8h]

31	0
NewDat[128:97]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 20-17. New Data Registers Field Descriptions

Bit	Name	Value	Description
31-0	NewDat[128:1]		New Data Bits (for all message objects).
		0	No new data has been written into the data portion of this message object by the Message Handler since the last time when this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.

20.3.14 Interrupt Pending X Register (DCAN INTPND X)

With the Interrupt Pending X Register, the CPU can detect if one or more bits in the different Interrupt Pending Registers are set. Each bit of this register represents a group of eight message objects. If at least one of the IntPnd bits of these message objects are set, the corresponding bit in the Interrupt Pending X Register will be set.

Figure 20-38. Interrupt Pending X Register (DCAN INTPND X) [offset = ACh]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntPndReg8		IntPndReg7		IntPndReg6		IntPndReg5		IntPndReg4		IntPndReg3		IntPndReg2		IntPndReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Example 2

Bit 0 of the Interrupt Pending X Register represents byte 0 of the Interrupt Pending 1 Register. If one or more bits in this byte are set, bit 0 of the Interrupt Pending X Register will be set.

20.3.15 Interrupt Pending Registers (DCAN INTPND12 to DCAN INTPND78)

These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after a reception or a successful transmission.

Figure 20-39. Interrupt Pending 12 Register [offset = B0h]

31	0
IntPnd[32:1]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-40. Interrupt Pending 34 Register [offset = B4h]

31	0
IntPnd[64:33]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-41. Interrupt Pending 56 Register [offset = B8h]

31	0
IntPnd[96:65]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-42. Interrupt Pending 78 Register [offset = BCh]

31	0
IntPnd[128:97]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 20-18. Interrupt Pending Registers Field Descriptions

Bit	Name	Value	Description
31-0	IntPnd[128:1]		Interrupt Pending Bits (for all message objects).
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt.

20.3.16 Message Valid X Register (DCAN MSGVAL X)

With the Message Valid X Register, the CPU can detect if one or more bits in the different Message Valid Registers are set. Each bit of this register represents a group of eight message objects. If at least one of the MsgVal bits of these message objects are set, the corresponding bit in the Message Valid X Register will be set.

Figure 20-43. Message Valid X Register (DCAN MSGVAL X) [offset = C0h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MsgValReg8		MsgValReg7		MsgValReg6		MsgValReg5		MsgValReg4		MsgValReg3		MsgValReg2		MsgValReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Example 3

Bit 0 of the Message Valid X Register represents byte 0 of the Message Valid 1 Register. If one or more bits in this byte are set, bit 0 of the Message Valid X Register will be set.

20.3.17 Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78)

These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after a reception or a successful transmission.

Figure 20-44. Message Valid 12 Register [offset = C4h]

31	0
MsgVal[32:1]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-45. Message Valid 34 Register [offset = C8h]

31	0
MsgVal[64:33]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-46. Message Valid 56 Register [offset = CCh]

31	0
MsgVal[96:65]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Figure 20-47. Message Valid 78 Register [offset = D0h]

31	0
MsgVal[128:97]	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 20-19. Message Valid Registers Field Descriptions

Bit	Name	Value	Description
31-0	MsgVal[128:1]		Message Valid Bits (for all message objects).
		0	This message object is ignored by the Message Handler.
		1	This message object is configured and will be considered by the Message Handler.

20.3.18 Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78)

The IntMux flag determines for each message object which of the two interrupt lines (DCAN0INT or DCAN1INT) will be asserted when the IntPnd of this message object is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN Control Register.

The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 Interface Register sets, or by Message Handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the Interrupt Register.

Figure 20-48. Interrupt Multiplexer 12 Register [offset = D8h]

31	0
IntMux[32:1]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 20-49. Interrupt Multiplexer 34 Register [offset = DCh]

31	0
IntMux[64:33]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 20-50. Interrupt Multiplexer 56 Register [offset = E0h]

31	0
IntMux[96:65]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 20-51. Interrupt Multiplexer 78 Register [offset = E4h]

31	0
IntMux[128:97]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 20-20. Interrupt Multiplexer Registers Field Descriptions

Bit	Name	Value	Description
31-0	IntMux[128:1]		Multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines. The mapping from the bits to the message objects is as follows: Bit 0 -> last implemented message object. Bit 1 -> message object number 1. Bit 2 -> message object number 2.
		0	DCAN0INT line is active if corresponding IntPnd flag is 1.
		1	DCAN1INT line is active if corresponding IntPnd flag is 1.

20.3.19 IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD)

The IF1/IF2 Command Register configure and Initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred.

A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to 1 to indicate that a transfer is in progress.

After 4 to 14 VBUS clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer concurs with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed.

NOTE: While the Busy bit is 1, IF1/IF2 Register sets are write protected.

If an invalid Message Number is written to bits [7:0] of the IF1/IF2 Command Register, the Message Handler may access an implemented (valid) message object instead.

Figure 20-52. IF1 Command Registers (DCAN IF1CMD) [offset = 100h]

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
WR_RD	Mask	Arb	Control	ClrIntPnd	TxRqst/NewDat	DataA	DataB
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15	14	13	12	11	10	9	8
Busy	Reserved	Reserved					
R-0	R/WP/C-0	R-0					

7	6	5	4	3	2	1	0
Message Number							
R/WP-1h							

LEGEND: R/W = Read/Write; R = Read; WP = Protected Write (protected by Busy bit); C = Clear by IF1 Access; -*n* = value after reset

Figure 20-53. IF1 Command Registers (CAN IF2CMD) [offset = 120h]

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
WR_RD	Mask	Arb	Control	ClrIntPnd	TxRqst/NewDat	DataA	DataB
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15	14	13	12	11	10	9	8
Busy	Reserved	Reserved					
R-0	R/WP/C-0	R-0					

7	6	5	4	3	2	1	0
Message Number							
R/WP-1h							

LEGEND: R/W = Read/Write; R = Read; WP = Protected Write (protected by Busy bit); C = Clear by IF1 Access; -*n* = value after reset

Table 20-21. IF1/IF2 Command Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	These bits are always read as 0. Writes have no effect.
23	WR_RD		Write/Read
		0	Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 register set.
		1	Direction = Write: Transfer direction is from the IF1/IF2 register set to the message object addressed by Message Number (Bits [7:0]).
22	Mask		Access Mask bits
		0	Mask bits will not be changed.
		1	Direction = Read: The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set. Direction = Write: The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).
21	Arb		Access Arbitration bits
		0	Arbitration bits will not be changed.
		1	Direction = Read: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set. Direction = Write: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).
20	Control		Access Control bits
		0	Control bits will not be changed.
		1	Direction = Read: The Message Control bits will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set. Direction = Write: The Message Control bits will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). If the TxRqst/NewDat bit in this register (Bit [18]) is set, the TxRqst/NewDat bit in the IF1/IF2 Message Control Register will be ignored.
19	ClrIntPnd		Clear Interrupt Pending bit
		0	IntPnd bit will not be changed.
		1	Direction = Read: Clears IntPnd bit in the message object. Direction = Write: This bit is ignored. Copying of IntPnd flag from IF1/IF2 Registers to Message RAM can be controlled by only the Control flag (Bit [20]).
18	TxRqst/NewDat		Access Transmission Request bit
		0	Direction = Read: NewDat bit will not be changed. Direction = Write: TxRqst/NewDat bit will be handled according to the Control bit.
		1	Direction = Read: Clears NewDat bit in the message object. Direction = Write: Sets TxRqst/NewDat in the message object. **Note:** If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to 1 and independent of the values in IF1/IF2 Message Control Register. A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/IF2 Message Control Register always reflect the status before resetting them.
17	DataA		Access Data Bytes 0-3
		0	Data Bytes 0-3 will not be changed.
		1	Direction = Read: The Data Bytes 0-3 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set. Direction = Write: The Data Bytes 0-3 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0]). **Note:** The duration of the message transfer is independent of the number of bytes to be transferred.

Table 20-21. IF1/IF2 Command Register Field Descriptions (continued)

Bit	Field	Value	Description
16	DataB		Access Data Bytes 4-7
		0	Data Bytes 4-7 will not be changed.
		1	Direction = Read: The Data Bytes 4-7 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set.
			Direction = Write: The Data Bytes 4-7 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0]).
			Note: The duration of the message transfer is independent of the number of bytes to be transferred.
15	Busy		Busy flag
		0	No transfer between IF1/IF2 Register set and Message RAM is in progress.
		1	Transfer between IF1/IF2 Register set and Message RAM is in progress. This bit is set to 1 after the message number has been written to bits [7:0]. IF1/IF2 Register set will be write-protected. The bit is cleared after read/write action has finished.
14	Reserved	0	Reserved. Do not use.
13-8	Reserved	0	These bits are always read as 0. Writes have no effect.
7-0	Message Number		Number of message object in Message RAM that is used for data transfer.
		0	Invalid message number.
		1h-10h	Valid message numbers for DCAN2.
		1h-20h	Valid message numbers for DCAN1.
		11h-FFh	Invalid message numbers for DCAN2.
		21h-FFh	Invalid message numbers for DCAN1.
			Note: When an invalid message number is written to the IF1/IF2 Command Register that is higher than the last implemented message object number, a modulo addressing will occur. For example, when accessing message object 33 in a DCAN module with 32 message objects only, the message object 1 will be accessed instead.

20.3.20 IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK)

The bits of the IF1/IF2 Mask Registers mirror the mask bits of a message object. The function of the relevant message objects bits is described in Section 20.2.5.1.

NOTE: While the Busy bit in the IF1/IF2 Command Register is 1, IF1/IF2 Register Set is write protected.

Figure 20-54. IF1 Mask Register (DCAN IF1MSK) [offset = 104h]

31	30	29	28	16
MXtd	MDir	Rsvd	Msk[28:16]	
R/WP-1	R/WP-1	R-1	R/WP-1FFFh	

15	0
Msk[15:0]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 20-55. IF2 Mask Register (DCAN IF2MSK) [offset = 124h]

31	30	29	28	16
MXtd	MDir	Rsvd	Msk[28:16]	
R/WP-1	R/WP-1	R-1	R/WP-1FFFh	

15	0
Msk[15:0]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 20-22. IF1/IF2 Mask Register Field Descriptions

Bit	Field	Value	Description
31	MXtd		Mask Extended Identifier
		0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.
			When 11-bit ("standard") identifiers are used for a message object, the identifiers of received Data Frames are written into bits ID[28:18]. For acceptance filtering, only these bits with mask bits Msk[28:18] are considered.
30	MDir		Mask Message Direction
		0	The message direction bit (Dir) has no effect on the acceptance filtering.
		1	The message direction bit (Dir) is used for acceptance filtering.
29	Reserved	1	These bits are always read as 1. Writes have no effect.
28-0	Msk[28:0]		Identifier Mask
		0	The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).
		1	The corresponding bit in the identifier of the message object is used for acceptance filtering.

20.3.21 IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB)

The bits of the IF1/IF2 Arbitration Registers mirror the arbitration bits of a message object. The function of the relevant message objects bits is described in Section 20.2.5.1.

NOTE: While the Busy bit in the IF1/IF2 Command Register is 1, IF1/IF2 Register Set is write protected.

Figure 20-56. IF1 Arbitration Register (DCAN IF1ARB) [offset = 108h]

31	30	29	28 ... 16
MsgVal	Xtd	Dir	ID[28:16]
R/WP-0	R/WP-0	R/WP-0	R/WP-0

15 ... 0
ID[15:0]
R/WP-0

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 20-57. IF2 Arbitration Register (DCAN IF2ARB) [offset = 128h]

31	30	29	28 ... 16
MsgVal	Xtd	Dir	ID[28:16]
R/WP-0	R/WP-0	R/WP-0	R/WP-0

15 ... 0
ID[15:0]
R/WP-0

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 20-23. IF1/IF2 Arbitration Register Field Descriptions

Bit	Field	Value	Description
31	MsgVal		Message Valid
		0	The message object is ignored by the Message Handler.
		1	The message object is used by the Message Handler.
			Note: The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation, see Section 20.2.7.6 and Section 20.2.7.7.
30	Xtd		Extended Identifier
		0	The 11-bit ("standard") identifier is used for this message object.
		1	The 29-bit ("extended") identifier is used for this message object.
29	Dir		Message direction
		0	Direction = Receive: On TxRqst, a Remote Frame with the identifier of this message object is transmitted. On receiving a Data Frame with a matching identifier, this message is stored in this message object.
		1	Direction = Transmit: On TxRqst, the respective message object is transmitted as a Data Frame. On receiving a Remote Frame with a matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
28-0	ID[28:0]		Message Identifier
		ID[28:0]	29-bit Identifier ("Extended Frame")
		ID[28:18]	11-bit Identifier ("Standard Frame")

The Arbitration bits ID[28:0], Xtd, and Dir are used to define the identifier and type of outgoing messages and (together with the Mask bits Msk[28:0], MXtd, and MDir) for acceptance filtering of incoming messages.

A received message is stored into the valid message object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame).

Extended frames can be stored only in message objects with Xtd = 1, standard frames in message objects with Xtd = 0.

If a received message (Data Frame or Remote Frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

20.3.22 IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL)

The bits of the IF1/IF2 Message Control Registers mirror the message control bits of a message object. The function of the relevant message objects bits is described in Section 20.2.5.1.

NOTE: While the Busy bit in the IF1/IF2 Command Register is 1, IF1/IF2 Register Set is write protected.

Figure 20-58. IF1 Message Control Register (DCAN IF1MCTL) [offset = 10Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	6	5	4	3	2	1	0
EoB	Reserved			DLC			
R/WP-0	R-0			R/WP-0			

LEGEND: R/W = Read/Write; R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 20-59. IF2 Message Control Register (DCAN IF2MCTL) [offset = 12Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	6	5	4	3	2	1	0
EoB	Reserved			DLC			
R/WP-0	R-0			R/WP-0			

LEGEND: R/W = Read/Write; R = Read; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 20-24. IF1/IF2 Message Control Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These bits are always read as 0. Writes have no effect.
15	NewDat		New Data
		0	No new data has been written into the data portion of this message object by the Message Handler since the last time this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.
14	MsgLst		Message Lost (only valid for message objects with direction = receive).
		0	No message lost since the last time when this bit was reset by the CPU.
		1	The Message Handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		Interrupt Pending
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt. The interrupt identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		Use Acceptance Mask
		0	Mask is ignored.
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering.
			If the UMask bit is set to 1, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to 1.
11	TxIE		Transmit Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
10	RxIE		Receive Interrupt Enable
		0	IntPnd will not be triggered after the successful reception of a frame.
		1	IntPnd will be triggered after the successful reception of a frame.
9	RmtEn		Remote Enable
		0	At the reception of a Remote Frame, TxRqst is not changed.
		1	At the reception of a Remote Frame, TxRqst is set.
8	TxRqst		Transmit Request
		0	This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and not yet done.
7	EoB		End of Block
		0	The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
		1	The message object is a single message object or the last message object in a FIFO Buffer block.
			Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to 1.
6-4	Reserved	0	These bits are always read as 0. Writes have no effect.
3-0	DLC		Data Length Code
		0-8	Data Frame has 0-8 data bytes.
		9-15	Data Frame has 8 data bytes.
			Note: The Data Length Code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.

20.3.23 IF1/IF2 Data A and Data B Registers (DCAN IF1DATA/DATB, DCAN IF2DATA/DATB)

The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order.

In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first

Figure 20-60. IF1 Data A Register (DCAN IF1DATA) [offset = 110h]

31	24	23	16
Data 3		Data 2	
R/WP-0		R/WP-0	

15	8	7	0
Data 1		Data 0	
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 20-61. IF1 Data B Register (DCAN IF1DATB) [offset = 114h]

31	24	23	16
Data 7		Data 6	
R/WP-0		R/WP-0	

15	8	7	0
Data 5		Data 4	
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 20-62. IF2 Data A Register (DCAN IF2DATA) [offset = 130h]

31	24	23	16
Data 3		Data 2	
R/WP-0		R/WP-0	

15	8	7	0
Data 1		Data 0	
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 20-63. IF2 Data B Register (DCAN IF2DATB) [offset = 134h]

31	24	23	16
Data 7		Data 6	
R/WP-0		R/WP-0	

15	8	7	0
Data 5		Data 4	
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

20.3.24 IF3 Observation Register (DCAN IF3OBS)

The IF3 register set can automatically be updated with received message objects without the need to Initiate the transfer from Message RAM by CPU (Additional information can be found in Section 20.2.5.1).

The observation flags (bits [4:0]) in the IF3 Observation register are used to determine which data sections of the IF3 Interface Register set have to be read. After all marked data sections are read, the DCAN is enabled to update the IF3 Interface Register set with new data.

Any access order of single bytes or half-words is supported. When using byte or half-word accesses, a data section is marked as completed, if all bytes are read.

NOTE: If IF3 Update Enable is used and no Observation flag is set, the corresponding message objects will be copied to IF3.

A write access to this register enables updating of IF3 Interface Register set with new data.

The status of the current read-cycle can be observed via status flags (Bits [12:8]).

Since there is no IF3 interrupt line available in this device, no interrupt will be generated by the IF3Upd flag.

With this, the observation status bits and the IF3Upd bit could be used by the application to realize the notification about new IF3 content in polling mode.

Figure 20-64. IF3 Observation Register (DCAN IF3OBS) [offset = 140h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
IF3Upd	Reserved		IF3SDB	IF3SDA	IF3SC	IF3SA	IF3SM
R-0	R-0		R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
Reserved			DataB	DataA	Ctrl	Arb	Mask
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 20-25. IF3 Observation Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These bits are always read as 0. Writes have no effect.
15	IF3Upd		IF3 Update Data
		0	No new data has been loaded since IF3 was last read.
		1	New data has been loaded since IF3 was last read.
14-13	Reserved	0	These bits are always read as 0. Writes have no effect
12	IF3SDB		IF3 Status of Data B read access.
		0	All Data B bytes are already read or are not marked to be read.
		1	Data B section still has data to read.
11	IF3SDA		IF3 Status of Data A read access.
		0	All Data A bytes are already read or are not marked to be read.
		1	Data A section still has data to read.
10	IF3SC		IF3 Status of Control bits read access.
		0	All Control section bytes are already read or are not marked to be read.
		1	Control section still has data to read.

Table 20-25. IF3 Observation Register Field Descriptions (continued)

Bit	Field	Value	Description
9	IF3SA		IF3 Status of Arbitration data read access.
		0	All Arbitration data bytes are already read or are not marked to be read.
		1	Arbitration section still has data to read.
8	IF3SM		IF3 Status of Mask data read access.
		0	All Mask data bytes are already read or are not marked to be read.
		1	Mask section still has data to read.
7-5	Reserved	0	These bits are always read as 0. Writes have no effect
4	DataB		Data B read observation.
		0	Data B section does not need to be read.
		1	Data B section has to be read to enable next IF3 update.
3	DataA		Data A read observation.
		0	Data A section does not need to be read.
		1	Data A section has to be read to enable next IF3 update.
2	Ctrl		Ctrl read observation.
		0	Ctrl section does not need to be read.
		1	Ctrl section has to be read to enable next IF3 update.
1	Arb		Arbitration data read observation.
		0	Arbitration data does not need to be read.
		1	Arbitration data has to be read to enable next IF3 update.
0	Mask		Mask data read observation.
		0	Mask data does not need to be read.
		1	Mask data has to be read to enable next IF3 update.

20.3.25 IF3 Mask Register (DCAN IF3MSK)

Figure 20-65. IF3 Mask Register (DCAN IF3MSK) [offset = 144h]

31	30	29	28	16
MXtd	MDir	Rsvd	Msk[28:16]	
R-1	R-1	R-1	R-1FFFh	

15	0
Msk[15:0]	
R-FFFFh	

LEGEND: R = Read; -*n* = value after reset

Table 20-26. IF3 Mask Register Field Descriptions

Bit	Field	Value	Description
31	MXtd		Mask Extended Identifier
		0	The extended identifier bit (IDE) has no effect on acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.
			Note: When 11-bit ("standard") identifiers are used for a message object, the identifiers of received Data Frames are written into bits ID[28:18]. For acceptance filtering, only these bits, together with mask bits Msk[28:18], are considered.
30	MDir		Mask Message Direction
		0	The message direction bit (Dir) has no effect on acceptance filtering.
		1	The message direction bit (Dir) is used for acceptance filtering.
29	Reserved	1	This bit is always read as 1. Writes have no effect.
28-0	Msk[28:0]		Identifier Mask
		0	The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).
		1	The corresponding bit in the identifier of the message object is used for acceptance filtering.

20.3.26 IF3 Arbitration Register (DCAN IF3ARB)

Figure 20-66. IF3 Arbitration Register (DCAN IF3ARB) [offset = 148h]

31	30	29	28	16
MsgVal	Xtd	R-0	ID[28:16]	
R-0	R-0	R-0	R-0	

15	0
ID[15:0]	
R-0	

LEGEND: R = Read; -*n* = value after reset

Table 20-27. IF3 Arbitration Register Field Descriptions

Bit	Field	Value	Description
31	MsgVal		Message Valid
		0	The message object is ignored by the Message Handler.
		1	The message object is to be used by the Message Handler.
			Note: The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation, see Section 20.2.7.6 and Section 20.2.7.7.
30	Xtd		Extended Identifier
		0	The 11-bit ("standard") identifier is used for this message object.
		1	The 29-bit ("extended") identifier is used for this message object.
29	Dir		Message direction
		0	Direction = Receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On receiving a data frame with a matching identifier, the message is stored in this message object.
		1	Direction = Transmit: On TxRqst, the respective message object is transmitted as a data frame. On receiving a remote frame with a matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
28-0	ID[28:0]		Message Identifier
		ID[28:0]	29-bit Identifier ("Extended Frame")
		ID[28:18]	11-bit Identifier ("Standard Frame")

20.3.27 IF3 Message Control Register (DCAN IF3MCTL)

Figure 20-67. IF3 Message Control Register (DCAN IF3MCTL) [offset = 14Ch]

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6		4	3			0
EoB	Reserved			DLC			
R-0	R-0			R-0			

LEGEND: R = Read; -*n* = value after reset

Table 20-28. IF3 Message Control Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These bits are always read as 0. Writes have no effect.
15	NewDat		New Data
		0	No new data has been written into the data portion of this message object by the Message Handler since the last time this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.
14	MsgLst		Message Lost (only valid for message objects with direction = receive).
		0	No message lost since the last time when this bit was reset by the CPU.
		1	The Message Handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		Interrupt Pending
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt. The interrupt identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		Use Acceptance Mask
		0	Mask is ignored.
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering.
			If the UMask bit is set to 1, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to 1.
11	TxIE		Transmit Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
10	RxIE		Receive Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
9	RmtEn		Remote Enable
		0	At the reception of a Remote Frame, TxRqst is not changed.
		1	At the reception of a Remote Frame, TxRqst is set.
8	TxRqst		Transmit Request
		0	This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and not yet done.

Table 20-28. IF3 Message Control Register Field Descriptions (continued)

Bit	Field	Value	Description
7	EoB		End of Block
		0	The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.
		1	The message object is a single message object or the last message object in a FIFO Buffer block.
			Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to 1.
6-4	Reserved	0	These bits are always read as 0. Writes have no effect.
3-0	DLC		Data Length Code
		0-8	Data Frame has 0-8 data bytes.
		9-15	Data Frame has 8 data bytes.
			Note: The Data Length Code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.

20.3.28 IF3 Data A and Data B Registers (DCAN IF3DATA/DATB)

The data bytes of CAN messages are stored in the IF3 registers in the following order.

In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 20-68. IF3 Data A Register (DCAN IF3DATA) [offset = 150h]

31	24	23	16
Data 3		Data 2	
R-0		R-0	

15	8	7	0
Data 1		Data 0	
R-0		R-0	

LEGEND: R = Read; -*n* = value after reset

Figure 20-69. IF3 Data B Register (DCAN IF3DATB) [offset = 154h]

31	24	23	16
Data 7		Data 6	
R/WP-0		R/WP-0	

15	8	7	0
Data 5		Data 4	
R-0		R-0	

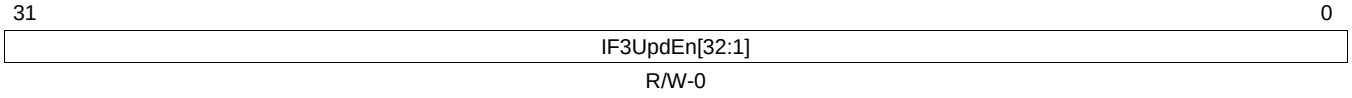
LEGEND: R = Read; -*n* = value after reset

20.3.29 IF3 Update Enable Registers (DCAN IF3UPD12 to IF3UPD78)

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (for example, due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set.

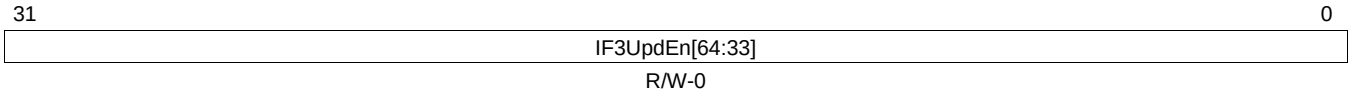
NOTE: IF3 Update enable should not be set for transmit objects.

Figure 20-70. IF3 Update Enable 12 Register [offset = 160h]



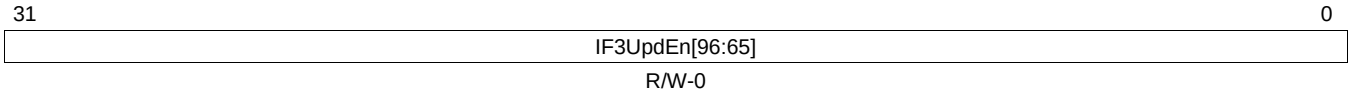
LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 20-71. IF3 Update Enable 34 Register [offset = 164h]



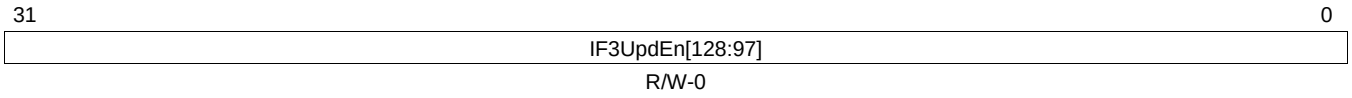
LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 20-72. IF3 Update Enable 56 Register [offset = 168h]



LEGEND: R/W = Read/Write; -*n* = value after reset

Figure 20-73. IF3 Update Enable 78 Register [offset = 16Ch]



LEGEND: R/W = Read/Write; -*n* = value after reset

Table 20-29. IF3 Update Control Register Field Descriptions

Bit	Name	Value	Description
31-0	IF3UpdEn[128:1]		IF3 Update Enabled (for all message objects).
		0	Automatic IF3 update is disabled for this message object.
		1	Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.

20.3.30 CAN TX IO Control Register (DCAN TIOC)

The CAN_TX pin of the DCAN module can be used as general-purpose IO pin if CAN function is not needed.

NOTE: The values of the IO Control registers are only writable if Init bit of CAN Control Register is set.

The OD, Func, Dir, and Out bits of the CAN TX IO Control register are forced to certain values when Init bit of CAN Control Register is reset (see bit descriptions).

Figure 20-74. CAN TX IO Control Register (DCAN TIOC) [offset = 1E0h]

31	19	18	17	16
Reserved		PU	PD	OD
R-0		R/W-D	R/W-D	R/WP-0

15	4	3	2	1	0
Reserved		Func	Dir	Out	In
R-0		R/WP-0	R/WP-0	R/WP-0	R-U

LEGEND: R/W = Read/Write; R = Read; WP = Protected Write (protected by Init bit); D = Device-dependent; U = Undefined; -*n* = value after reset

Table 20-30. CAN TX IO Control Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	These bits are always read as 0. Writes have no effect.
18	PU		CAN_TX Pullup/Pulldown select. This bit is only active when CAN_TX is configured to be an input.
		0	CAN_TX Pulldown is selected, when pull logic is active (PD = 0).
		1	CAN_TX Pullup is selected, when pull logic is active (PD = 0).
17	PD		CAN_TX pull disable. This bit is only active when CAN_TX is configured to be an input.
		0	CAN_TX pull is active.
		1	CAN_TX pull is disabled.
16	OD		CAN_TX open drain enable. This bit is only active when CAN_TX is configured to be in GIO mode (TIOC.Func = 0).
		0	The CAN_TX pin is configured in push/pull mode.
		1	The CAN_TX pin is configured in open drain mode.
			Forced to 0 if Init bit of CAN control register is reset.
15-4	Reserved	0	These bits are always read as 0. Writes have no effect.
3	Func		CAN_TX function. This bit changes the function of the CAN_TX pin.
		0	CAN_TX pin is in GIO mode.
		1	CAN_TX pin is in functional mode (as an output to transmit CAN data).
			Forced to 1 if Init bit of CAN control register is reset.
2	Dir		CAN_TX data direction. This bit controls the direction of the CAN_TX pin when it is configured to be in GIO mode only (TIOC.Func = 0).
		0	The CAN_TX pin is an input.
		1	The CAN_TX pin is an output.
			Forced to 1 if Init bit of CAN control register is reset.
1	Out		CAN_TX data out write. This bit is only active when CAN_TX pin is configured to be in GIO mode (TIOC.Func = 0) and configured to be an output pin (TIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_TX pin.
		0	The CAN_TX pin is driven to logic low (0).
		1	The CAN_TX pin is at logic high (1).
			Forced to Tx output of the CAN Core, if Init bit of CAN Control register is reset.

Table 20-30. CAN TX IO Control Register Field Descriptions (continued)

Bit	Field	Value	Description
0	In		CAN_TX data in.
		0	The CAN_TX pin is at logic low (0).
		1	The CAN_TX pin is at logic high (1).
			Note: When CAN_TX pin is connected to a CAN transceiver, an external pullup resistor has to be used to ensure that the CAN bus will not be disturbed (for example, while the DCAN module is reset).

20.3.31 CAN RX IO Control Register (DCAN RIOC)

The CAN_RX pin of the DCAN module can be used as general-purpose IO pin if CAN function is not needed.

NOTE: The values of the IO Control registers are writable only if Init bit of CAN Control Register is set.

The OD, Func, and Dir bits of the CAN RX IO Control register are forced to certain values when Init bit of CAN Control Register is reset, see bit description.

Figure 20-75. CAN RX IO Control Register (DCAN RIOC) [offset = 1E4h]

31	19	18	17	16
Reserved		PU	PD	OD
R-0		R/W-D	R/W-D	R/WP-0

15	4	3	2	1	0
Reserved		Func	Dir	Out	In
R-0		R/WP-0	R/WP-0	R/WP-0	R-U

LEGEND: R/W = Read/Write; R = Read; WP = Protected Write (protected by Init bit); D = Device-dependent; U = Undefined; -*n* = value after reset

Table 20-31. CAN RX IO Control Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	These bits are always read as 0. Writes have no effect.
18	PU		CAN_RX Pullup/Pulldown select. This bit is only active when CAN_RX is configured to be an input.
		0	CAN_RX Pulldown is selected, when pull logic is active (PD = 0).
		1	CAN_RX Pullup is selected, when pull logic is active (PD = 0).
17	PD		CAN_RX pull disable. This bit is only active when CAN_RX is configured to be an input.
		0	CAN_RX pull is active.
		1	CAN_RX pull is disabled.
16	OD		CAN_RX open drain enable. This bit is only active when CAN_RX is configured to be in GIO mode (RIOC.Func = 0).
		0	The CAN_RX pin is configured in push/pull mode.
		1	The CAN_RX pin is configured in open drain mode.
			Forced to 0 if Init bit of CAN control register is reset.
15-4	Reserved	0	These bits are always read as 0. Writes have no effect.
3	Func		CAN_RX function. This bit changes the function of the CAN_RX pin.
		0	CAN_RX pin is in GIO mode.
		1	CAN_RX pin is in functional mode (as an input to receive CAN data).
			Forced to 1 if Init bit of CAN control register is reset.

Table 20-31. CAN RX IO Control Register Field Descriptions (continued)

Bit	Field	Value	Description
2	Dir		CAN_RX data direction. This bit controls the direction of the CAN_RX pin when it is configured to be in GIO mode only (RIOC.Func = 0).
		0	The CAN_RX pin is an input.
		1	The CAN_RX pin is an output.
			Forced to 0 if Init bit of CAN control register is reset.
1	Out		CAN_RX data out write. This bit is only active when CAN_RX pin is configured to be in GIO mode (RIOC.Func = 0) and configured to be an output pin (RIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_RX pin.
		0	The CAN_RX pin is driven to logic low (0).
		1	The CAN_RX pin is at logic high (1).
0	In		CAN_RX data in.
		0	The CAN_RX pin is at logic low (0).
		1	The CAN_RX pin is at logic high (1).
			Note: When CAN_RX pin is connected to a CAN transceiver, an external pullup resistor has to be used to ensure that the CAN bus will not be disturbed (for example, while the DCAN module is reset).

Multi-Buffered Serial Peripheral Interface Module (MibSPI)

This chapter provides the specifications for a 16-bit configurable synchronous multi-buffered multi-pin serial peripheral interface (MibSPI). The MibSPI is a programmable-length shift register used for high-speed communication between external peripherals or other microcontrollers.

Throughout this chapter, all references to SPI also apply to MibSPI unless otherwise noted.

Topic **Page**

21.1 Overview

The MibSPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (2 to 16 bits) to be shifted in and out of the device at a programmed bit-transfer rate. Typical applications for the SPI include interfacing to external peripherals, such as I/Os, memories, display drivers, and analog-to-digital converters.

The SPI has the following attributes:

- 16-bit shift register
- Receive buffer register
- 11-bit baud clock generator
- Serial clock (SPICLK) pin
- SPI enable ($\overline{\text{SPIENA}}$) pin
- Up to 4 slave chip select ($\overline{\text{SPICS}}$) pins
- SPICLK can be internally-generated (and driven) or received from an external clock source
- Each word transferred can have a unique format
- SPI pins can be used as functional or digital Input/Output pins (GIOs)

NOTE: SIMO - Slave In Master Out Pin
SOMI - Slave Out Master In Pin
SPICS - SPI Chip Select Pin
SPIENA - SPI Enable Pin.

This device contains one MibSPI (MibSPI1) and two SPIs (SPIP2 and SPI3). All three modules support up to four SPICS, one SPICLK pins. MibSPI1 and SPI3 support one SPIENA pin each. All of them support one SPISOMI/SPISIMO pair.

21.1.1 Word Format Options

Each word transferred can have a unique format. Several format characteristics are programmable for each word transferred:

- SPICLK frequency
- Character length (2 to 16 bits)
- Phase (with and without delay)
- Polarity (high or low)
- Parity enabled/disabled
- Chip Select (CS) timers for setup and hold
- Shift direction (Most Significant Bit (MSB)- first or Least Significant Bit (LSB)- first)

21.1.2 Multi-buffering (Mib) Support

The MibSPI has a programmable buffer memory that enables programmed transmission to be completed without CPU intervention. The buffers are combined in different Transfer Groups (TGs) that can be triggered by external events (timers, Input/Output activity, and so on) or by the internal tick counter. The internal tick counter supports periodic trigger events.

21.1.2.1 Multi-buffer Mode

Multi-buffer mode is an extension to the SPI. In multi-buffer mode, many extended features are configurable:

- Number of buffers for each peripheral (or data source/destination, up to 128 buffers supported) or group (up to 8 groupings)
- Triggers for each groups trigger types and trigger sources for individual groups (14 external trigger sources and 1 internal trigger source supported)
- Memory fault detection via an internal parity circuit

21.1.2.2 Compatibility Mode

Compatibility mode of the MibSPI makes it behave exactly like a standard platform SPI module and ensures full compatibility with other SPIs. All features in compatibility mode of the MibSPI are directly applicable to a SPI. Multi-buffer mode features are not available in compatibility mode.

NOTE: The SPIDAT0 register is not accessible in the multi-buffer mode of MibSPI. It is only accessible in compatibility mode.

21.1.3 Transmission Lock (Multi-Buffer Mode Master Only)

Some slave devices require transmission of a command followed by data. In this case the SPI transaction should not be interrupted by another group transfer. The LOCK bit within each buffer allows a consecutive transfer to happen without being interrupted by another higher-priority group transfer.

21.2 Operating Modes

The SPI can be configured via software to operate as either a master or a slave. The MASTER bit (SPIGCR1[0]) selects the configuration of the SPISIMO and SPISOMI pins. CLKMOD bit (SPIGCR1[1]) determines whether an internal or external clock source will be used.

The slave chip select ($\overline{\text{SPICS}}$) pins are used when communicating with multiple slave devices or, with a single slave, to delimit messages containing a leading register address. When the a write occurs to SPIDAT1 in master mode, the $\overline{\text{SPICS}}$ pins are automatically driven to select the specified slave.

Handshaking mechanism, provided by the $\overline{\text{SPIENA}}$ pin, enables a slave SPI to delay the generation of the clock signal supplied by the master if it is not prepared for the next exchange of data.

21.2.1 Pin Configurations

The SPI supports data connections as shown in Table 21-1.

NOTE:

1. When the SPICS signals are disabled, the chip select field in the transmit data is not used.
2. When the SPIENA signal is disabled, the $\overline{\text{SPIENA}}$ pin is ignored in master mode, and not driven as part of the SPI transaction in slave mode.

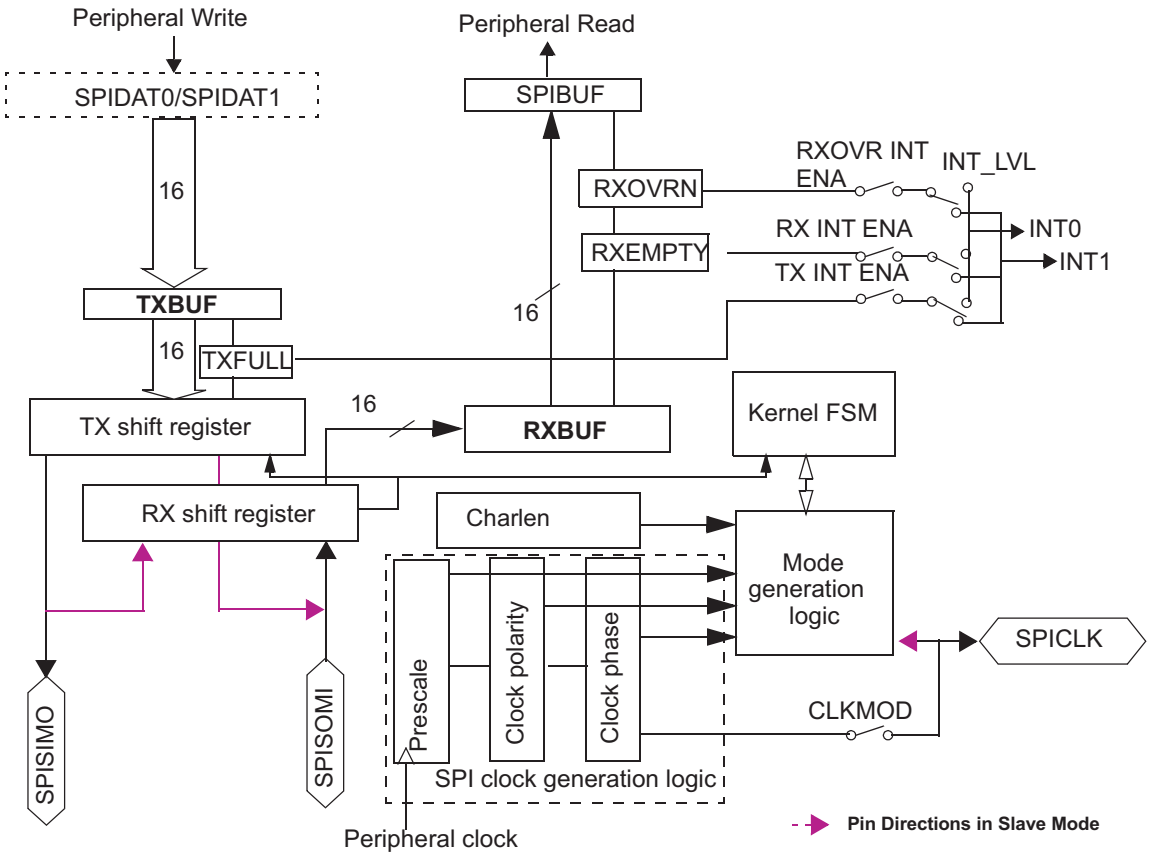
Table 21-1. Pin Configurations

Pin	Master Mode		Slave Mode	
SPICLK	Drives the clock to external devices		Receives the clock from the external master	
SPISOMI	Receives data from the external slave		Sends data to the external master	
SPISIMO	Transmits data to the external slave		Receives data from the external master	
$\overline{\text{SPIENA}}$	**SPIENA disabled:** GIO	**SPIENA enabled:** Receives ENA signal from the external slave	**SPIENA disabled:** GIO	**SPIENA enabled:** Drives ENA signal from the external master
$\overline{\text{SPICS}}$	**SPICS disabled:** GIO	**SPICS enabled:** Selects one or more slave devices	**SPICS disabled:** GIO	**SPICS enabled:** Receives the CS signal from the external master

21.2.2 Data Handling

Figure 21-1 shows the SPI transaction hardware. TXBUF and RXBUF are internal buffers that are intended to improve the overall throughput of data transfer. TXBUF is a transmit buffer, while RXBUF is a receive buffer.

Figure 21-1. SPI Functional Logic Diagram



1 This is a representative diagram, which shows three-pin mode hardware.
2 TXBUF, RXBUF, and SHIFT_REGISTER are user-invisible registers.
3 SPIDAT0 and SPIDAT1 are user-visible, and are physically mapped to the contents of TXBUF.
4 SPISIMO, SPISOMI, SPICLK pin directions depend on the Master or Slave Mode.

21.2.2.1 Data Sequencing when SPIDAT0 or SPIDAT1 is Written

- If both the TX shift register and TXBUF are empty, then the data is directly copied to the TX shift register. If transmit interrupts are enabled, a transmitter-empty interrupt is generated.
- If the TX shift register is already full or is in the process of shifting and if TXBUF is empty, then the data written to SPIDAT0 / SPIDAT1 is copied to TXBUF and TXFULL flag is set to 1 at the same time.
- When a shift operation is complete, data from the TXBUF (if it is full) is copied into TX shift register and the TXFULL flag is cleared to 0 to indicate that next data can be fetched. A transmitter-empty interrupt (if enabled) is generated at the same time.

21.2.2.2 Data Sequencing when All Bits Shifted into RXSHIFT Register

- If both SPIBUF and RXBUF are empty, the received data in RX shift register is directly copied into SPIBUF and the receive interrupt (if enabled) is generated. The RXEMPTY flag in SPIBUF is cleared at the same time.
- If SPIBUF is already full at the end of receive completion, the RX shift register contents is copied to RXBUF. The receive complete interrupt line remains high.
- If SPIBUF is read by the CPU and if RXBUF is full, then the contents of RXBUF are copied to SPIBUF as soon as SPIBUF is read. RXEMPTY flag remains cleared, indicating that SPIBUF is still full.
- If both SPIBUF and RXBUF are full, then RXBUF will be overwritten and the RXOVR interrupt flag is set and an interrupt is generated, if enabled.

21.2.2.3 Three-Pin Mode

In master mode configuration (MASTER = 1 and CLKMOD = 1), the SPI provides the serial clock on the SPICLK pin. Data is transmitted on the SPISIMO pin and received on the SPISOMI pin (see Figure 21-2).

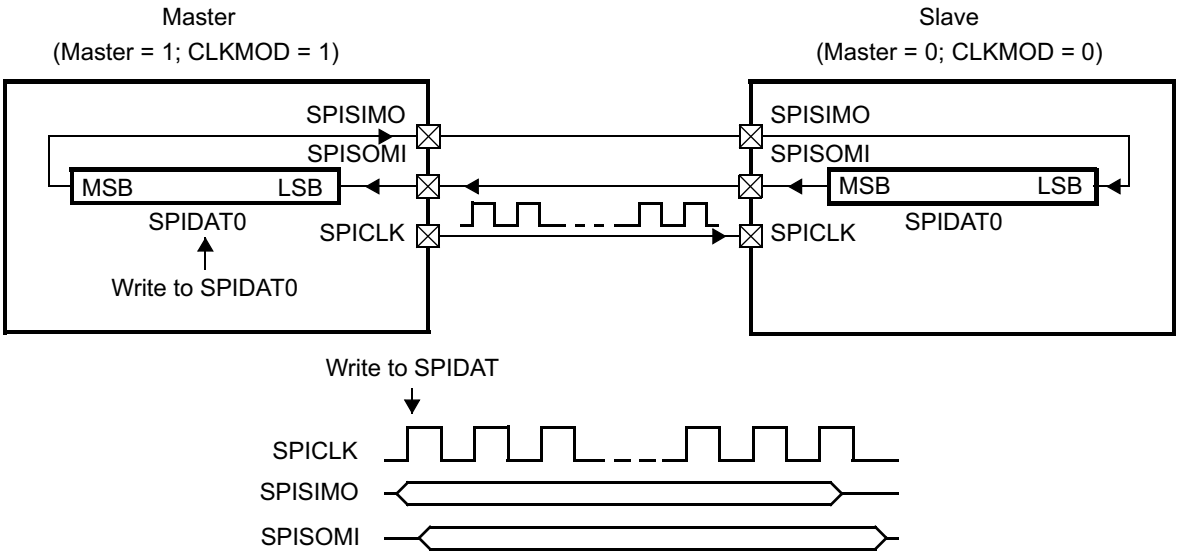
Data written to the shift register (SPIDAT0 / SPIDAT1) initiates data transmission on the SPISIMO pin, MSB first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB of the SPIDAT0 register. When the selected number of bits have been transmitted, the received data in the shift register is transferred to the SPIBUF register for the CPU to read. Data is stored right-justified in SPIBUF.

See Section 21.2.2.1 and Section 21.2.2.2 for details about the data handling for transmit and receive operations.

In slave mode configuration (MASTER = 0 and CLKMOD = 0), data shifts out on the SPISOMI pin and in on the SPISIMO pin. The SPICLK pin is used as the input for the serial shift clock, which is supplied from the external network master. The transfer rate is defined by this clock.

Data written to the SPIDAT0 or SPIDAT1 register is transmitted to the network when the SPICLK signal is received from the network master. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts data on the SPISIMO pin into the RX shift register. If data is to be transmitted by the slave simultaneously, it must be written to the SPIDAT0 or SPIDAT1register before the beginning of the SPICLK signal.

Figure 21-2. SPI Three-Pin Operation



21.2.3 Operation with $\overline{\text{SPICS}}$

In master mode, each chip select signal is used to select a specific slave. In slave mode, the chip select signal is used to enable and disable the transfer. Chip-select functionality is enabled by setting one of the $\overline{\text{SPICS}}$ pins as a chip select. It is disabled by setting all $\overline{\text{SPICS}}$ pins as GIOs in SPIPC0.

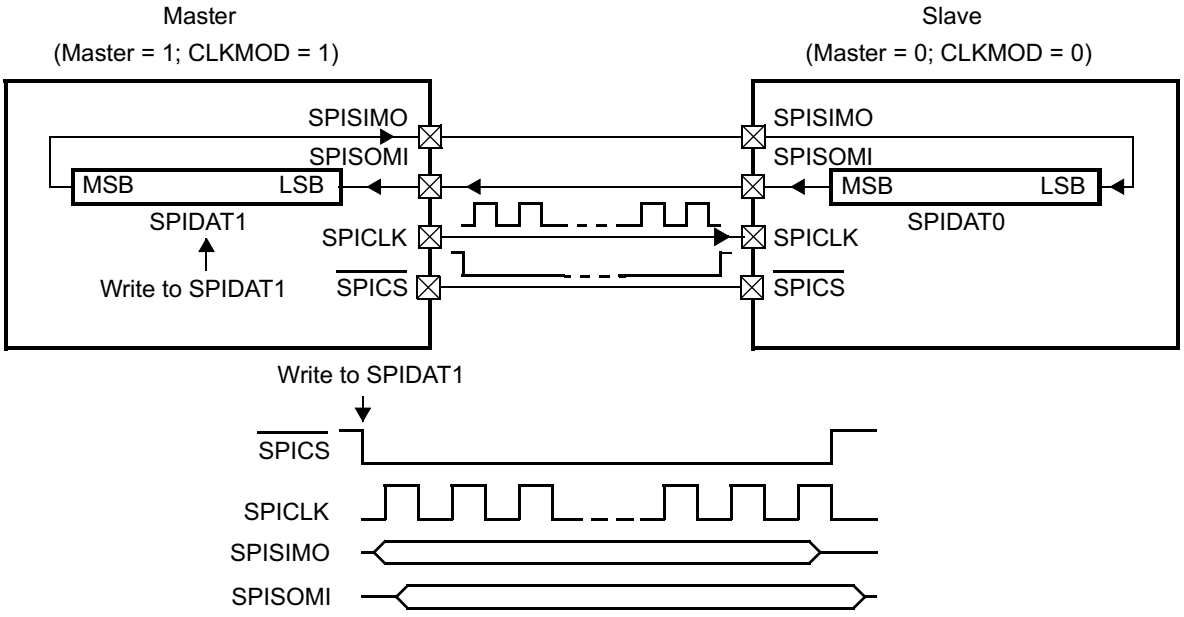
21.2.3.1 Multiple Chip Selects

The $\overline{\text{SPICS}}$ pins that are used must be configured as functional pins in the SPIPC0 register. The default pattern to be put on the $\overline{\text{SPICS}}$ when all the slaves are deactivated is set in the SPIDEF register. This pattern allows different slaves with different chip-select polarity to be activated by the SPI.

The master-mode SPI is capable of driving either 0 or 1 as the active value for any $\overline{\text{SPICS}}$ output pin. The drive state for $\overline{\text{SPICS}}$ pins is controlled by the CSNR field of SPIDAT1. The pattern that is driven will select the slave to which the transmission is dedicated.

In slave mode, the SPI can only be selected by an active value of 0 on any of its selected $\overline{\text{SPICS}}$ input pins.

Figure 21-3. Operation with $\overline{\text{SPICS}}$



21.2.4 Operation with $\overline{\text{SPIENA}}$

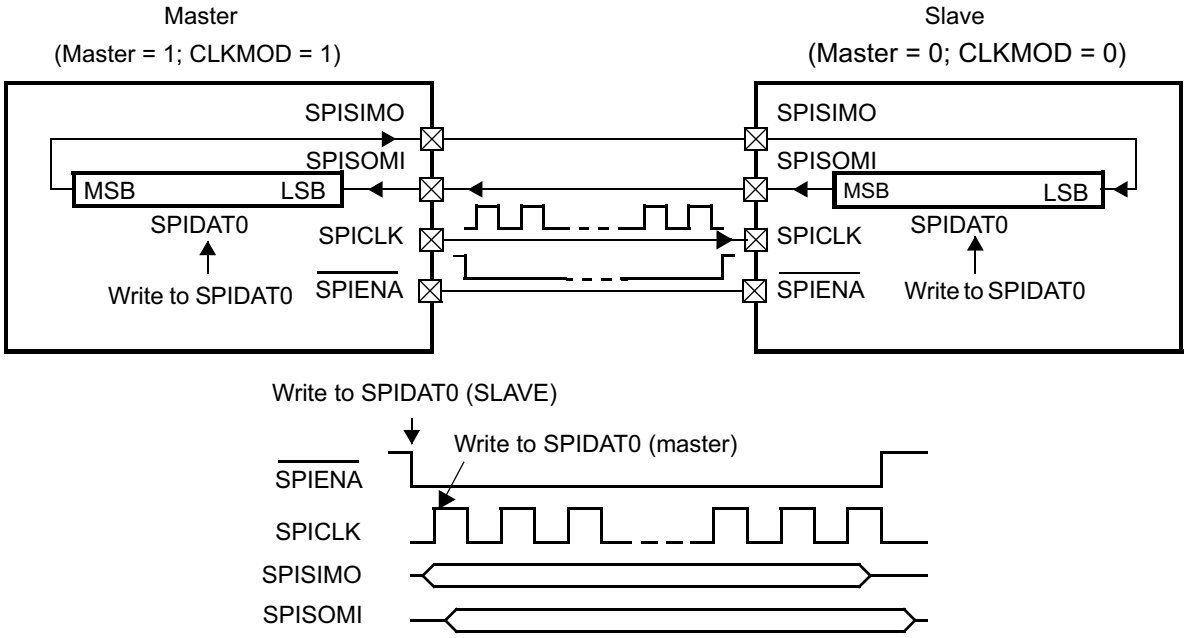
The $\overline{\text{SPIENA}}$ operates as a WAIT signal pin. For both the slave and the master, the $\overline{\text{SPIENA}}$ pin must be configured to be functional (SPIPC0[8] = 1). In this mode, an active low signal from the slave on the $\overline{\text{SPIENA}}$ pin allows the master SPI to drive the clock pulse stream. A high signal tells the master to hold the clock signal (and delay SPI activity).

If the $\overline{\text{SPIENA}}$ pin is in high-impedance mode (ENABLE_HIGHZ = 1), the slave will put $\overline{\text{SPIENA}}$ into the high-impedance once it completes receiving a new character. If the $\overline{\text{SPIENA}}$ pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave will drive $\overline{\text{SPIENA}}$ to 1 once it completes receiving a new character. The slave will drive $\overline{\text{SPIENA}}$ low again for the next word to transfer, after new data is written to the slave TX shift register.

In master mode (CLKMOD = 1), if the $\overline{\text{SPIENA}}$ pin is configured as functional, then the pin will act as an input pin. If configured as a slave SPI and as functional, the $\overline{\text{SPIENA}}$ pin acts as an output pin.

NOTE: During a transfer, if a slave-mode SPI detects a deassertion of its chip select before its internal character length counter overflows, then it places SPISOMI and $\overline{\text{SPIENA}}$ (if ENABLE_HIGHZ bit is set to 1) in high-z mode. Once this condition has occurred, if a SPICLK edge is detected while the chip select is deasserted, then the SPI stops that transfer and sets an DLENERR error flag and generates an interrupt (if enabled).

Figure 21-4. Operation with $\overline{\text{SPIENA}}$



21.2.5 Five-Pin Operation (Hardware Handshaking)

Five-pin operation combines the functionality of three-pin mode, plus the enable pin and one or more chip select pins. The result is full hardware handshaking. To use this mode, both the SPIENA pin and the required number of SPICS pins must be configured as functional pins.

If the SPIENA pin is in high-impedance mode (ENABLE_HIGHZ = 1), the slave SPI will put this signal into the high-impedance state by default. The slave will drive the signal SPIENA low when new data is written to the slave shift register and the slave has been selected by the master (SPICS is low).

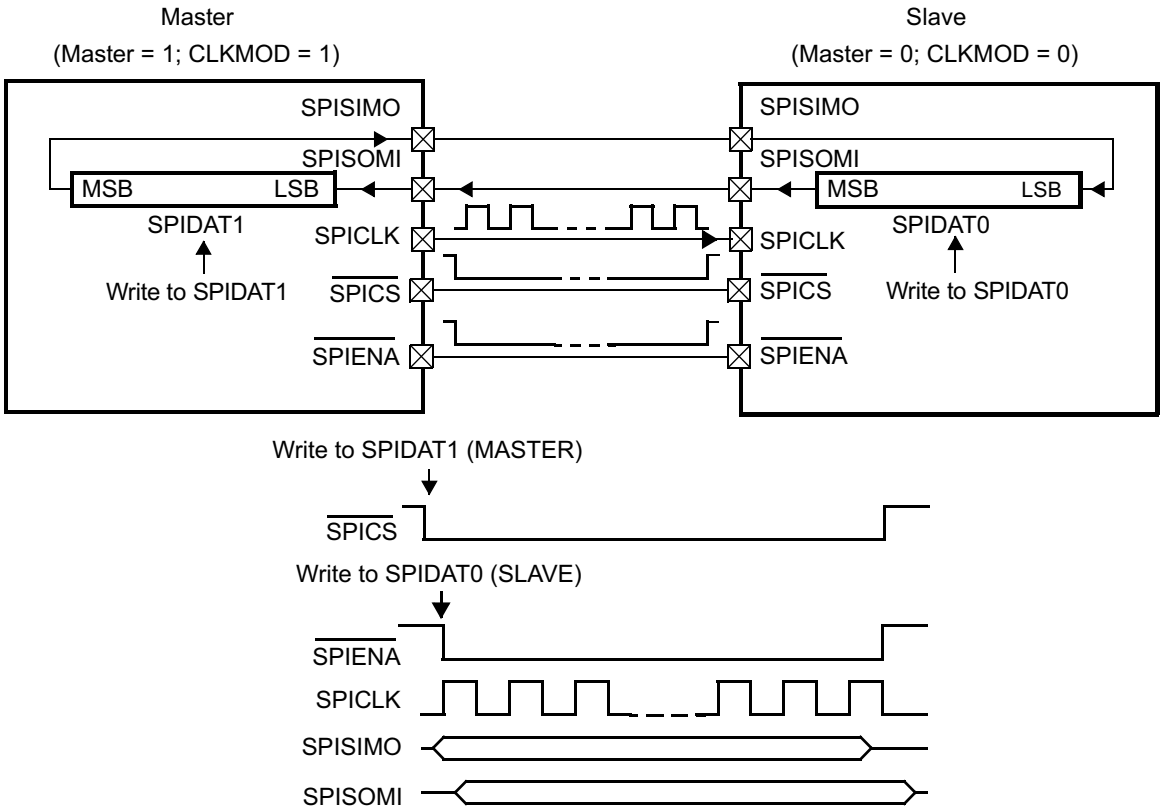
If the SPIENA pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave SPI drives this pin high by default when it is in functional mode. The slave SPI will drive the SPIENA signal low when new data is written to the slave shift register (SPIDAT0/SPIDAT1) and the slave is selected by the master (SPICS is low). If the slave is deselected by the master (SPICS goes high), the slave SPIENA signal is driven high.

NOTE: Push-pull mode of the SPIENA pin can be used only when there is a single slave in the system. When multiple SPI slave devices are connected to the common SPIENA pin, all of the slaves should configure their SPIENA pins in high-z mode.

In master mode, if the SPICS pins are configured as functional pins, then the pins will be in output mode. A write to the master's SPIDAT1/SPIDAT0 register will automatically drive the SPICS signals low. The master will drive the SPICS signals high again after completing the transfer of the bits of the data.

In slave mode (CLKMOD = 0), the SPICS pins will act as SPI functional inputs.

Figure 21-5. SPI Five-Pin Option with SPIENA and SPICS



21.2.6 Data Formats

To support multiple different types of slaves in one SPI network, four independent data word formats are implemented that allow configuration of individual data word length, polarity, phase, and bit rate. Each word transmitted can select which data format to use via the bits DFSEL[1:0] in its control field from one of the four data word formats. Same data format can be supported on multiple chip selects.

Data formats 0, 1, 2, and 3 can be configured through SPIFMTx control registers.

Each SPI data format includes the standard SPI data format with enhanced features:

- Individually-configurable shift direction can be used to select MSB first or LSB first, whereas the position of the MSB depends on the configured data word length.
- Receive data is automatically right-aligned, independent of shift direction and data word length. Transmit data has to be written right-aligned into the SPI and the internal shift register will transmit according to the selected shift direction and data word length for correct transfer.
- To increase fault detection of data transmission and reception, an odd or even parity bit can be added at the end of a data word. The parity generator can be enabled or disabled individually for each data format. If a received parity bit does not match with the locally calculated parity bit, the parity error flag (PARITYERR) is set and an interrupt is asserted (if enabled).

Since the master-mode SPI can drive two consecutive accesses to the same slave, an 8-bit delay counter is available to satisfy the delay time for data to be refreshed in the accessed slave. The delay counter can be programmed as part of the data format.

CHARLEN[4:0] specifies the number of bits (2 to 16) in the data word. The CHARLEN[4:0] value directs the state control logic to count the number of bits received or transmitted to determine when a complete word is transferred.

Data word length **must** be programmed to the same length for both the **master** and the **slave**. However, when chip selects are used, there may be multiple targets with different lengths in the system.

NOTE: Data must be right-justified when it is written to the SPI for transmission irrespective of its character length or word length.

Figure 21-6 shows how a 12-bit word (0xEC9) needs to be written to the transmit buffer to be transmitted correctly.

Figure 21-6. Format for Transmitting an 12-Bit Word

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	x	1	1	1	0	1	1	0	0	1	0	0	1

NOTE: The received data is always stored right-justified regardless of the character length or direction of shifting and is padded with leading 0s when the character length is less than 16 bits.

Figure 21-7 shows how a 10-bit word (0x0A2) is stored in the buffer once it is received.

Figure 21-7. Format for Receiving an 10-Bit Word

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0

21.2.7 Clocking Modes

SPICLK may operate in four different modes, depending on the choice of phase (delay/no delay) and the polarity (rising edge/falling edge) of the clock.

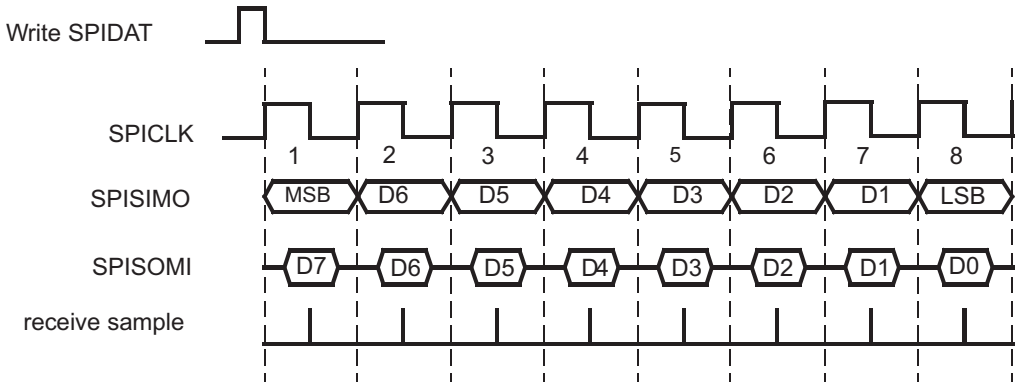
The data input and output edges depend on the values of both POLARITY and PHASE as shown in Table 21-2.

Table 21-2. Clocking Modes

POLARITY	PHASE	Action
0	0	Data is output on the rising edge of SPICLK. Input data is latched on the falling edge.
0	1	Data is output one half-cycle before the first rising edge of SPICLK and on subsequent falling edges. Input data is latched on the rising edge of SPICLK.
1	0	Data is output on the falling edge of SPICLK. Input data is latched on the rising edge.
1	1	Data is output one half-cycle before the first falling edge of SPICLK and on subsequent rising edges. Input data is latched on the falling edge of SPICLK.

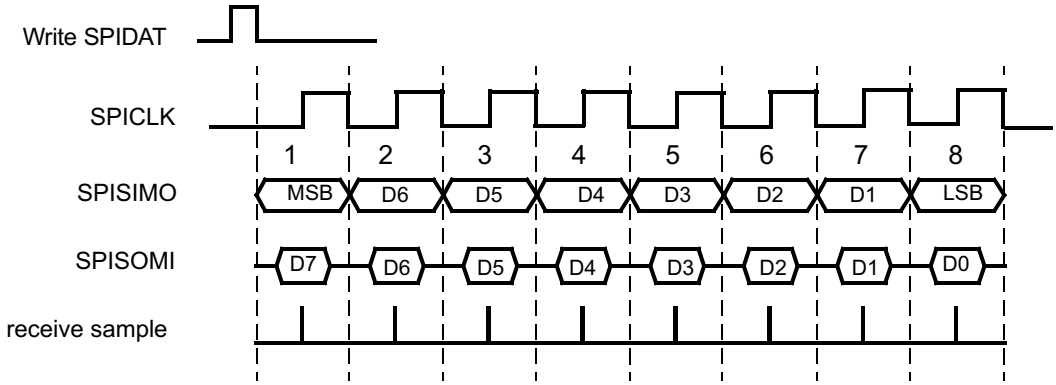
Figure 21-8 to Figure 21-11 illustrate the four possible configurations of **SPICLK** corresponding to each mode. Having four signal options allows the SPI to interface with many different types of serial devices.

Figure 21-8. Clock Mode with Polarity = 0 and Phase = 0



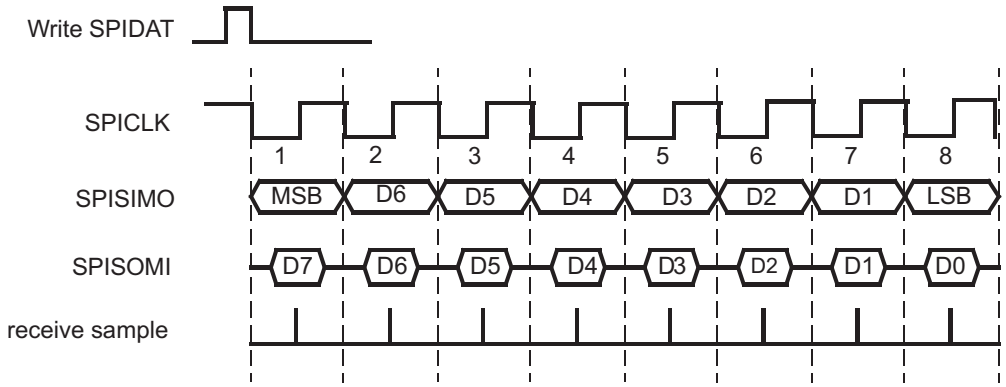
Data is output on the rising edge of SPICLK.
Input data is latched on the falling edge of SPICLK.

Figure 21-9. Clock Mode with Polarity = 0 and Phase = 1



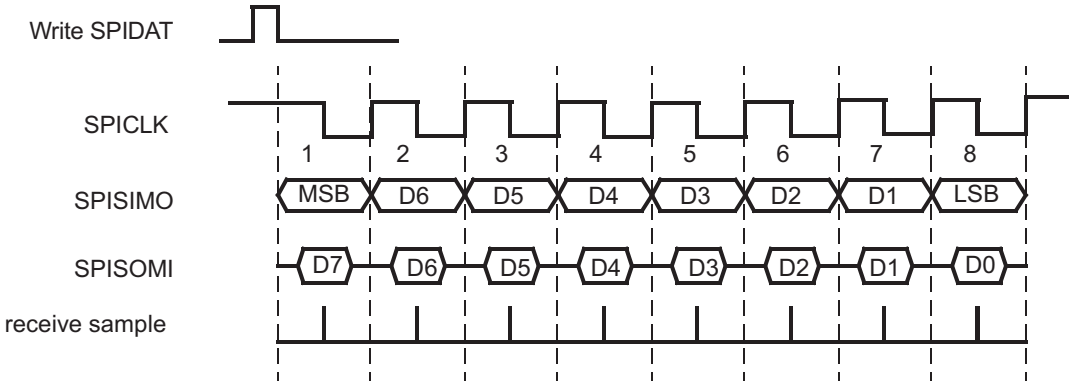
Data is output one-half cycle before the first rising edge of SPICLK and on subsequent falling edges of SPICLK
Input data is latched on the rising edge of SPICLK

Figure 21-10. Clock Mode with Polarity = 1 and Phase = 0



Data is output on the falling edge of SPICLK.
Input data is latched on the rising edge of SPICLK.

Figure 21-11. Clock Mode with Polarity = 1 and Phase = 1

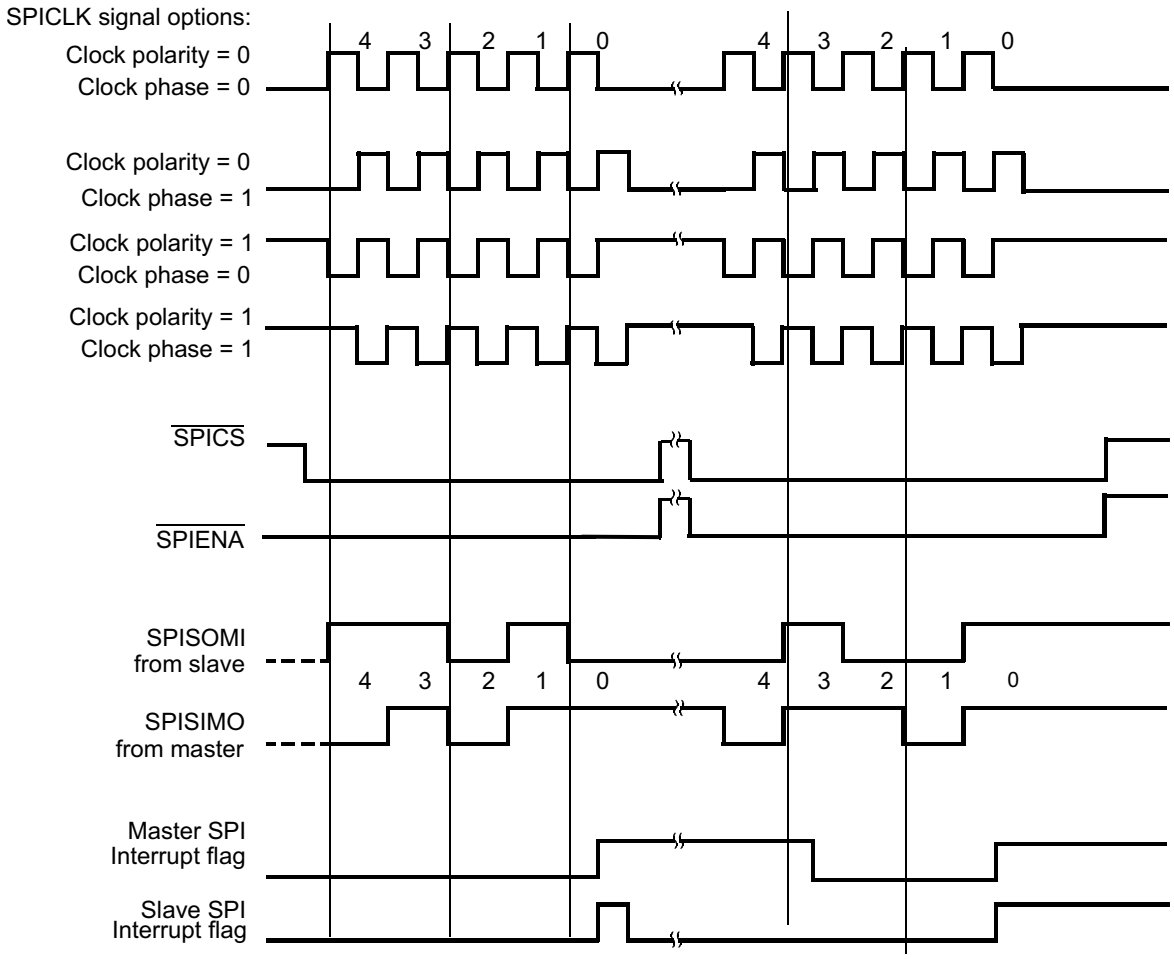


Data is output one-half cycle before the first falling edge of SPICLK and on the subsequent rising edges of SPICLK.
Input data is latched on the falling edge of SPICLK.

21.2.8 Data Transfer Example

Figure 21-12 illustrates a SPI data transfer between two devices using a character length of five bits.

Figure 21-12. Five Bits per Character (5-Pin Option)



21.2.9 Decoded and Encoded Chip Select (Master Only)

In this device the SPI can connect to up to 4 individual slave devices using chip-selects by routing one wire to each slave. The 4 chip selects in the control field are directly connected to the 4 pins. The default value of each chip select (not active) can be configured via the register CSDEF. During a transmission, the value of the chip select control field (CSNR) of the SPIDAT1 register (SPIDAT1) is driven on the $\overline{\text{SPICS}}$ pins. When the transmission finishes the default chip-select value (defined by the CSDEF register) is put on the $\overline{\text{SPICS}}$ pins.

The SPI can support more than 4 slaves by using encoded chip selects. To connect the SPI with encoded slaves devices, the CSNR field allows multiple active $\overline{\text{SPICS}}$ pins at the same time, which enables encoded chip selects from 0 to 16. To use encoded chip selects, all 4 chip select lines have to be connected to each slave device and each slave needs to have a unique chip-select address. The CSDEF register is used to provide the address at which slaves devices are all de-selected.

Users can combine decoded and encoded chip selects. For example, *n* $\overline{\text{SPICS}}$ pins can be used for encoding a *n*-bit address and the remaining pins can be connected to decoded-mode slaves.

21.2.10 Variable Chip Select Setup and Hold Timing (Master Only)

In order to support slow slave devices a delay counter can be configured to delay data transmission after the chip select is activated. A second delay counter can be configured to delay the chip select deactivation after the last data bit is transferred. Both delay counters are clocked with peripheral clock (VCLK).

If a particular data format specifically does not require these additional set-up or hold times for the chip select pin(s), then they can be disabled in the corresponding SPIFMTx register.

21.2.11 Hold Chip-Select Active

Some slave devices require the chip select signal to be held continuously active during several consecutive data word transfers. Other slave devices require the chip select signal to be deactivated between consecutive data word transfers.

CSHOLD is programmable in both master and slave modes of the multi-buffer mode of SPI. However, the meaning of CSHOLD in master mode and slave mode are different.

NOTE: If the CSHOLD bit is set within the current data control field, the programmed hold time and the following programmed set-up time will not be applied between transactions.

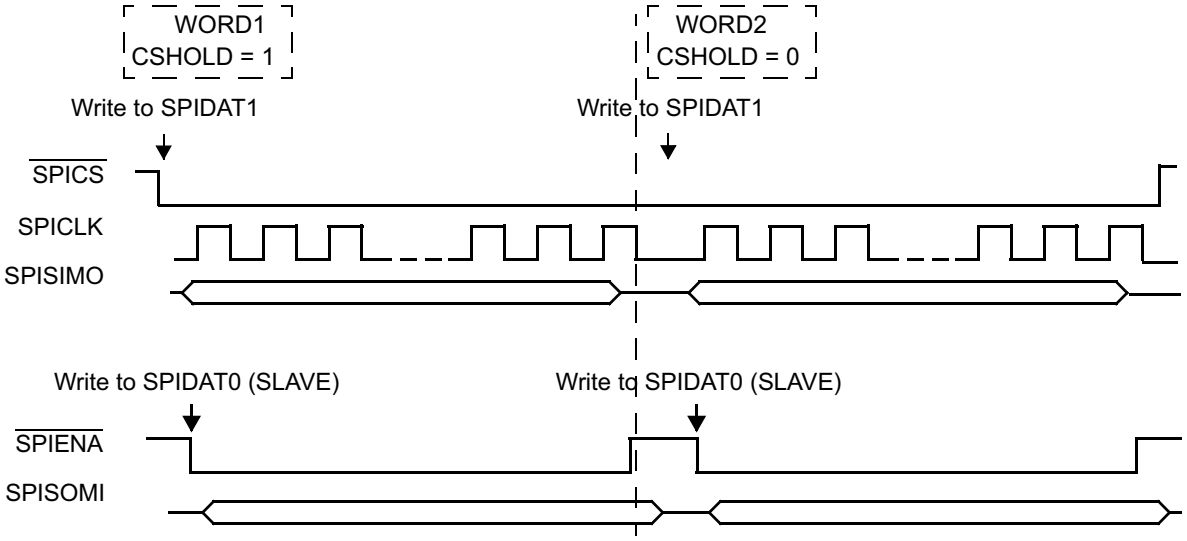
21.2.11.1 The CSHOLD Bit in Master Mode

Each word in a master-mode SPI can be individually initialized for one of the two modes via the CSHOLD bit in its control field.

If the CSHOLD bit is set in the control field of a word, the chip select signal will not be deactivated until the next control field is loaded with new chip select information. Since the chip-select is maintained active between two transfers, the chip-select hold delay (T2CDELAY) is not applied at the end of the current transaction, and the chip-select set-up time delay (C2TDELAY) is not applied as well at the beginning of the following transaction. However, the wait delay (WDELAY) will be still applied between the two transactions, if the WDEL bit is set within the control field.

Figure 21-13 shows the SPI pins when a master-mode SPI transfers a word that has its CSHOLD bit set. The chip-select pins will not be deasserted after the completion of this word. If the next word to transmit has the same chip-select number (CSNR) value, the chip select pins will be maintained until the completion of the second word, regardless of whether the CSHOLD bit is set or not.

Figure 21-13. Typical Diagram when a Buffer in Master is in CSHOLD Mode (SPI-SPI)



21.2.11.2 The CSHOLD Bit in Slave Mode (Multi-buffered Mode)

If the CSHOLD bit in a buffer is set to 1, then the MibSPI does not wait for the $\overline{\text{SPICS}}$ pins to be de-activated at the end of the shift operation to copy the received data to the receive RAM. With this feature, it is possible for a slave in multi-buffer mode to do multiple data transfers without requiring the $\overline{\text{SPICS}}$ pins to be deasserted between two buffer transfers.

If the CSHOLD bit in a buffer is cleared to 0 in a slave MibSPI, even after the shift operation is done, the MibSPI waits until the $\overline{\text{SPICS}}$ pin (if functional) is deasserted to copy the received data to the RXRAM.

If the CSHOLD bit is maintained as 0 across all the buffers, then the slave in multi-buffer mode requires its $\overline{\text{SPICS}}$ pins to be deasserted between any two buffer transfers; otherwise, the Slave SPI will be unable to respond to the next data transfer.

NOTE: In compatibility mode, the slave does not require the $\overline{\text{SPICS}}$ pin to be deasserted between two buffer transfers. The CSHOLD bit of the slave will be ignored in compatibility mode.

21.2.12 Detection of Slave Desynchronization (Master Only)

When a slave supports generation of an enable signal (ENA), desynchronization can be detected. With the enable signal a slave indicates to the master that it is ready to exchange data. A desynchronization can occur if one or more clock edges are missed by the slave. In this case the slave may block the SOMI line until it detects clock edges corresponding to the next data word. This would corrupt the data word of the desynchronized slave and the consecutive data word. A configurable 8-bit time-out counter (T2EDELAY), which is clocked with SPICLK, is implemented to detect this slave malfunction. After the transmission has finished (end of last bit transferred: either last data bit or parity bit) the counter is started. If the ENA signal generated by the slave does not become inactive before the counter overflows, the DESYNC flag is set and an interrupt is asserted (if enabled).

NOTE: Inconsistency of Desynchronization Flag in Compatibility Mode MibSPI

Because of the nature of this error, under some circumstances it is possible for a desync error detected for the previous buffer to be visible in the current buffer. This is due to the fact that receive completion flag/interrupt will be generated when the buffer transfer is completed. But desync will be detected after the buffer transfer is completed. So, if VBUS master reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition. This inconsistency in the desync flag is valid only in compatibility mode of MibSPI. In multi-buffer mode, the desync flag is always assured to be for the current buffer.

21.2.13 ENA Signal Time-Out (Master Only)

The SPI in master mode waits for the hardware handshake signal (ENA) coming from the addressed slave before performing a data transfer. To avoid stalling the SPI by a non-responsive slave device, a time-out value can be configured using C2EDELAY. If the time-out counter overflows before an active ENA signal is sampled, the TIMEOUT flag in the status register SPIFLG is set and the TIMEOUT flag in the status field of the corresponding buffer is set.

NOTE: When the chip select signal becomes active, no breaks in transmission are allowed. The next arbitration is performed while waiting for the time-out to occur.

21.2.14 Data-Length Error

A SPI can generate an error flag by detecting any mismatch in length of received or transmitted data and the programmed character length under certain conditions.

Data-Length Error in Master Mode: During a data transfer, if the SPI detects a de-assertion of the $\overline{\text{SPIENA}}$ pin (by the slave) while the character counter is not overflowed, then an error flag is set to indicate a data-length error. This can be caused by a slave receiving extra clocks (for example, due to noise on the SPICLK line).

NOTE: In a master mode SPI, the data length error will be generated only if the $\overline{\text{SPIENA}}$ pin is enabled as a functional pin.

Data-Length Error in Slave Mode: During a transfer, if the SPI detects a de-assertion of the $\overline{\text{SPICS}}$ pin before its character length counter overflows, then an error flag is set to indicate a data-length error. This situation can arise If the slave SPI misses one or more SPICLK pulses from the master. This error in slave mode implies that both the transmitted and received data were not complete.

NOTE: In a slave-mode SPI, the data-length error flag will be generated only if at least one of the $\overline{\text{SPICS}}$ pins are configured as functional, and are being used for selecting the slave.

21.2.15 Continuous Self-Test (Master/Slave)

During data transfer, the SPI compares its own internal transmit data with its transmit data on the bus. The sample point for the compare is at one-half SPI clock after transmit point. If the data on the bus does not match the expected value, the bit-error (BITERR) flag is set and an interrupt is asserted if enabled.

NOTE: The compare is made from the output pin using its input buffer.

21.2.16 Half Duplex Mode

SPI by protocol is Full Duplex in nature, which means simultaneous TX and RX operations happen on two separate data pins, SIMO and SOMI. However, it is possible to use SPI/MibSPI to do the TX-only operation (ignoring the RX data) and the RX-only operation (using dummy TX data and ignoring the TX pin). But this will require that both SOMI and SIMO lines are bonded out in a chip to be able to support both TX-only or RX-only features.

21.2.16.1 Half Duplex Mode in Master

The Half Duplex Mode gives an additional flexibility to use the SIMO pin that is normally used TX pin in Master mode to work like an RX pin while the HDUPLEX_ENAx bit is set to 1. In Half Duplex Master mode, SIMO pin itself acts as an RX pin. Switching between Full Duplex and Half Duplex can be achieved using the SPIFMTx registers being selected using the DFSEL[1:0] bits of SPIDATx or TXRAM locations.

21.2.16.2 Half Duplex Mode in Slave

In case of Slave mode, the SIMO pin which is normally an RX pin, will act as a TX pin while the HDUPLEX_ENAx bit is set to 1. In Half Duplex Slave mode, SIMO pin itself acts as a TX pin. Switching between Full Duplex and Half Duplex can be achieved using the SPIFMTx registers being selected using the DFSEL[1:0] bits of SPIDATx or TXRAM locations.

21.3 Test Features

21.3.1 Internal Loop-Back Test Mode (Master Only)

The internal loop-back self-test mode can be utilized to test the SPI transmit and receive paths, including the shift registers, the SPI buffer registers, and the parity generator. In this mode the transmit signal is internally feedback to the receiver, whereas the SIMO, SOMI, and CLK pin are disconnected; that is, the transmitted data is internally transferred to the corresponding receive buffer while external signals remain unchanged.

This mode allows the CPU to write into the transmit buffer, and check that the receive buffer contains the correct transmit data. If an error occurs the corresponding error is set within the status field.

NOTE: This mode cannot be changed during transmission.

21.3.2 Input/Output Loopback Test Mode

Input/Output Loopback Test mode supports the testing of all Input/Output pins without the aid of an external interface. Loopback can be configured as either analog-loopback (loopback through the pin-level input/output buffers) or digital loopback (internal to the SPI module). With Input/Output Loopback, all functional features of the SPI can be tested. Transmit data is fed back through the receive-data line(s). See Figure 21-14 for a diagram of the types of feedback available. The IOLPBKTSTCR register defines all of the available control fields.

In loopback mode, it is also possible to induce various error conditions. See Section 21.8.39 for details of the register field controlling these features.

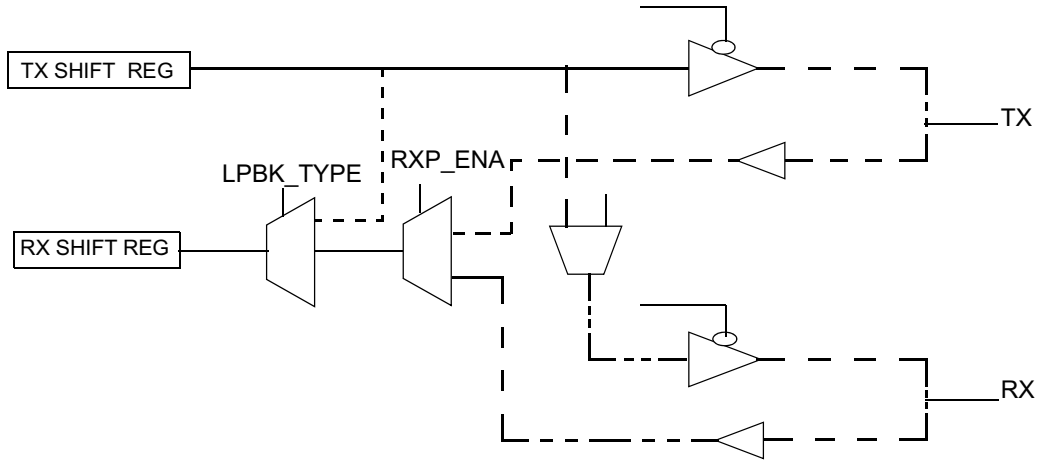
In Input/Output loopback test modes, even when the module is in slave mode, the SPICLK is generated internally. This SPICLK is used for all loopback-mode SPI transactions. Slave-mode features can be tested without the help of another master SPI, using the internally-generated SPICLK. Chip selects are also generated by the slave itself while it is in Input/Output loopback mode.

In Input/Output loopback test modes, if the module is in master mode, the nENA signal is also generated by internal logic so that an external interface is not required.

NOTE: Usage Guideline for Input/Output Loopback

Input/Output Loopback mode should be used with caution because, in some configurations, even the receive pins will be driven with transmit data. During testing, it should be ensured that none of the SPI pins are driven by any other device connected to them. Otherwise, if analog loopback is selected in I/O Loopback mode, then testing may damage the device.

Figure 21-14. I/O Paths during I/O Loopback Modes



Checks the analog loopback path through the receive buffer

Checks the analog loopback path through the transmit buffer

Digital loopback path

This diagram is intended to illustrate loopback paths and therefore may omit some normal-mode paths.

21.3.2.1 IO Loopback Mode Operation in Slave Mode

In multi-buffer slave mode, there are some additional requirements for using I/O loopback mode (IOLPBK). In multi-buffer slave mode, the chip-select pins are the triggers for various TGs. Enabling the IOLPBK mode by writing 0xA to the IOLPBTSTENA bits of the IOLPBKTSTCR register triggers TG0 by driving $\overline{\text{SPICS}}$ to 0. The actual number of chip selects can be programmed to have any or all of the $\overline{\text{SPICS}}$ pins as functional. All other configurations should be completed before enabling the IOLPBK mode in multi-buffer slave mode since it triggers TG0.

After the first buffer transfer is completed, the CSNR field of the current buffer is used to trigger the next buffer. So, if multiple TGs are desired to be tested, then the CSNR field of the final buffer in each TG should hold the number of the next TG to be triggered. As long as TG boundaries are well defined and are enabled, the completion of one TG will trigger the next TG.

To stop the transfer in multi-buffer slave mode in I/O Loopback configuration, either IOLPBK mode can be disabled by writing 5h to the IOLPBTSTENA bits or all of the TGs can be disabled.

21.4 General-Purpose I/O

All of the SPI pins may be programmed via the SPIPCx control registers to be either functional or general-purpose I/O pins.

If the SPI function is to be used, application software must ensure that at least the SPICLK pin and the SOMI and/or SIMO pins are configured as SPI functional pins, and not as GIO pins, or else the SPI state machine will be held in reset, preventing SPI transactions.

SPI pins support:

- internal pull-up resistors
- internal pull-down resistors
- open-drain or push-pull mode

21.5 Low-Power Mode

The SPI can be put into either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SPI. During global low-power mode, all clocks to the SPI are turned off, making the module completely inactive.

Local low-power mode is asserted by setting the POWERDOWN (SPIGCR1[8]) bit; setting this bit stops the clocks to the SPI internal logic and registers. Setting the POWERDOWN bit causes the SPI to enter local low-power mode and clearing the POWERDOWN bit causes SPI to exit from local low-power mode. All registers remain accessible during local power-down mode, since the clock to the SPI registers is temporarily re-enabled for each access. RAM buffers are also accessible during low power mode.

NOTE: Since entering a low-power mode has the effect of suspending all state-machine activities, care must be taken when entering such modes to ensure that a valid state is entered when low-power mode is active. Application software must ensure that a low power mode is not entered during a data transfer.

21.6 Interrupts

There are two levels of vectorized interrupts supported by the SPI. These interrupts can be caused under the following circumstances:

- Transmission error
- Receive overrun
- Receive complete (receive buffer full)
- Transmit buffer empty

These interrupts may be enabled or disabled via the SPIINT0 register.

During transmission, if one of the following errors occurs: BITERR, DESYNC, DLENERR, PARITYERR, or TIMEOUT, the corresponding bit in the SPIFLG register is set. If the corresponding enable bit is set, then an interrupt is generated. The level of all the above interrupts is set by the bit fields in the SPILVL register.

The error interrupts are enabled and prioritized independently from each other, but the interrupt generated will be the same if multiple errors are enabled on the same level. The SPIFLG register should be used to determine the actual cause of an error.

NOTE: Since there are two interrupt lines, one each for Level 0 and Level 1, it is possible for a programmer to separate out the interrupts for receive buffer full and transmit buffer empty. By programming one to Level 0 and the other to Level 1, it is possible to avoid a check on whether an interrupt occurred for transmit or for receive. A programmer can also choose to group all of the error interrupts into one interrupt line and both TX-empty and RX-full interrupts into another interrupt line using the LVL control register. In this way, it is possible to separate error-checking from normal data handling.

21.6.1 Interrupts in Multi-Buffer Mode

In multi-buffer mode, the SPI can generate interrupts on two levels.

In normal multi-buffer operation, the receive and transmit are not used and therefore the enable bits of SPIINT0 are not used.

The interrupts available in multi-buffer mode are:

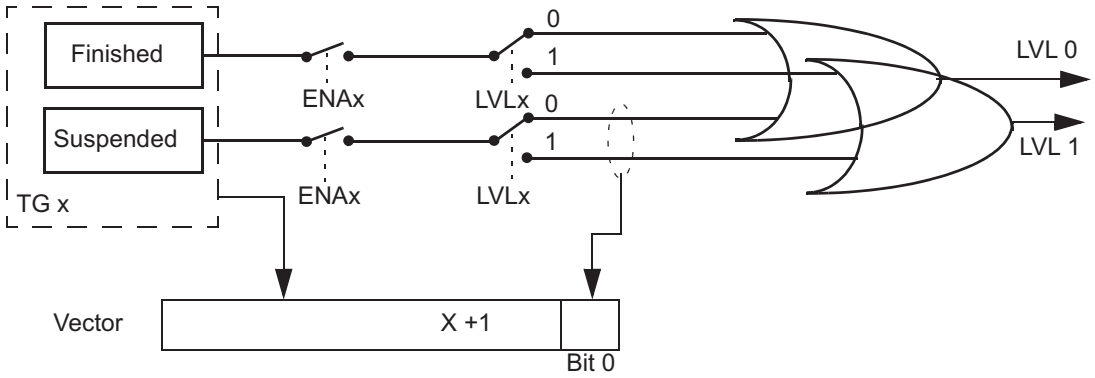
- Transmission error interrupt
- Receive overrun interrupt
- TG suspended interrupt
- TG completed interrupt

When a TG has finished and the corresponding enable bit in the TGINTENA register is set, a transfer-finished interrupt is generated. The level of priority of the interrupt is determined by the corresponding bit in the TGINTLVL register.

When a TG is suspended by a buffer that has been set as suspend to wait until TXFULL flag or/and RXEMPTY flag are set, and if the corresponding bit in the TGINTENA register is set, an transfer-suspended interrupt is generated. The level of priority of the interrupt is determined by the corresponding bit in the TGINTLVL register.

Figure 21-15 illustrates the TG interrupts.

Figure 21-15. TG Interrupt Structure

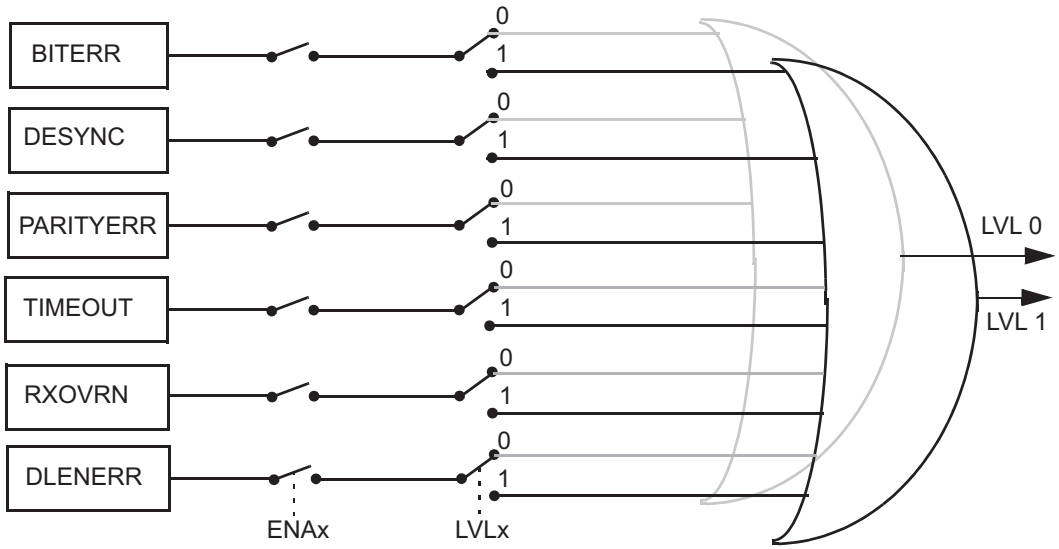


During transmission, if one of the following errors occurs, BITERR, DESYNC, PARITYERR, TIMEOUT, DLENERR, the corresponding flag in the SPIFLG register is set. If the enable bit is set, then an interrupt is generated. The level of the interrupts could be generated according to the bit field in SPILVL register.

The RXOVRN interrupt is generated when a buffer in the RXRAM is overwritten by a new received word. While writing newly received data to a RXRAM location, if the RXEMPTY bit of the corresponding location is 0, then the RXOVR bit will be set to 1 during the write operation, so that the buffer starts to indicate an overrun. This RXOVR flag is also reflected in SPIFLG register as RXOVRNINTFLG and the corresponding vector number is updated in TGINTVECT0/TGINTVECT1 register. If an overrun interrupt is enabled, then an interrupt will be generated indicating an overrun condition.

The error interrupts are enabled and prioritized independently from each other, but the vector generated by the SPI will be the same if multiple errors are enabled on the same level.

Figure 21-16. SPIFLG Interrupt Structure



Since the priority of an error interrupt is lower than a completion/suspend interrupt for a TG, the interrupts can be split into two levels. By programming all the error interrupts into Level 0 and TG-complete / TG-suspend interrupts into Level 1, it is possible to get a clear indication of the source of error interrupts. However, when a vector register shows an error interrupt, the actual buffer for which the error has occurred is not readily identifiable. Since each buffer in the multi-buffer RAM is stored along with its individual status flags, each buffer should be read until a buffer with any error flag set is found.

A separate interrupt line is provided to indicate the uncorrectable error condition in the MibSPI. This line is available (and valid) only in the multi-buffer mode of the MibSPI module and if the parity error detection feature for multi-buffer RAM is enabled.

21.7 Module Configuration

MibSPI can be configured to function as normal SPI and multi-buffered SPI. Upon power-up or a system-level reset, each bit in the module registers is set to a default state. The registers are writable only after the RESET bit is set to 1.

21.7.1 Compatibility (SPI) Mode Configuration

The following list details the configuration steps that software should perform prior to the transmission or reception of data. As long as the SPIEN bit in the Global Control Register 1 (SPIGCR1) is cleared to 0 the entire time that the SPI is being configured, the order in which the registers are programmed is not important.

- Enable SPI by setting RESET bit.
- Configure the SIMO, SOMI, SPICLK, and optional $\overline{\text{SPICS}}$ and $\overline{\text{SPIENA}}$, pins for SPI functionality by setting the corresponding bit in SPIPC0 register.
- Configure the module to function as Master or Slave using CLKMOD and MASTER bits.
- Configure the required SPI data format using SPIFMTx register.
- If the module is selected to function as Master, the delay parameters can be configured using SPIDELAY register.
- Enable the Interrupts using SPIINT0 register if required.
- Select the chip select to be used by setting CSNR bits in SPIDAT1 register.
- Configure CSHOLD and WDEL bits in SPIDAT1 register if required.
- Select the Data word format by setting DFSEL bits. Select the Number of the configured SPIFMTx register (0 to 3) to used for the communication.
- Set LOOPBACK bit to connect the transmitter to the receiver internally. (This feature is used to perform a self-test. Do not configure for normal communication to external devices).
- Set SPIENA to 1 after the SPI is configured.
- Perform Transmit and receive data, using SPIDAT1 and SPIBUF register.
- You must wait for TXFULL to reset or TXINT before writing next data to SPIDAT1 register.
- You must wait for RXEMPTY to reset or RXINT before reading the data from SPIBUF register.

21.7.2 MibSPI Mode Configuration

The following list details the configuration steps that software should perform prior to the transmission or reception of data in MIBSPI mode. As long as the SPIEN bit in the Global Control Register 1 (SPIGCR1) is cleared to 0 the entire time that the SPI is being configured, the order in which the registers are programmed is not important.

- Enable SPI by setting RESET bit.
- Set MSPIENA to 1 to get access to multi-buffer mode registers.
- Configure the SIMO, SOMI, SPICLK, and optional $\overline{\text{SPICS}}$ and $\overline{\text{SPIENA}}$, pins for SPI functionality by setting the corresponding bit in SPIPC0 register.
- Configure the module to function as Master or Slave using CLKMOD and MASTER bits.
- Configure the required SPI data format using SPIFMTx register.
- If the module is selected to function as Master, the delay parameters can be configured using SPIDELAY register.
- Check for BUFINITACTIVE bit to be active before configuring MIBSPI RAM. (From Device Power On it take Number of Buffers × Peripheral clock period to initialize complete RAM.)
- Enable the Transfer Group interrupts using TGITENST register if required.
- Enable error interrupts using SPIINT0 register if required.
- Set SPIENA to 1 after the SPI is configured.
- The Trigger Source, Trigger Event, Transfer Group start address for the corresponding Transfer groups can be configured using the corresponding TGxCTRL register.
- Configure LPEND to specify the end address of the last TG.
- Similar to SPIDAT1 register 16 Bit Control fields in every TXRAM buffer in the TG has to be configured.
- Configure one of the eight BUFMODE available for each buffer.
- Fill the data to be transmitted in TXDATA field in TXRAM buffers.
- Configure TGENA bit to enable the required Transfer groups. (In case of Trigger event always setting TGENA will trigger the transfer group).
- At the occurrence of the correct trigger event the Transfer group will be triggered and data gets transmitted and received one after the other with out any CPU intervention.
- You can poll Transfer-group interrupt flag or wait for a transfer-completed interrupt to read and write new data to the buffers.

21.8 Control Registers

This section describes the SPI control, data, and pin registers. The registers support 8-bit, 16-bit and 32-bit writes. The offset is relative to the associated base address of this module in a system. The base address for the control registers is FFF7 F400h for MibSPI1, FFF7 F600h for SPI2, and FFF7 F800h for SPI3.

NOTE: TI highly recommends that write values corresponding to the reserved locations of registers be maintained as 0 consistently. This allows future enhancements to use these reserved bits as control bits without affecting the functionality of the module with any older versions of software.

Table 21-3. SPI Registers

Offset	Acronym	Register Description	Section
00h	SPIGCR0	SPI Global Control Register 0	Section 21.8.1
04h	SPIGCR1	SPI Global Control Register 1	Section 21.8.2
08h	SPIINT0	SPI Interrupt Register	Section 21.8.3
0Ch	SPILVL	SPI Interrupt Level Register	Section 21.8.4
10h	SPIFLG	SPI Flag Register	Section 21.8.5
14h	SPIPC0	SPI Pin Control Register 0	Section 21.8.6
18h	SPIPC1	SPI Pin Control Register 1	Section 21.8.7
1Ch	SPIPC2	SPI Pin Control Register 2	Section 21.8.8
20h	SPIPC3	SPI Pin Control Register 3	Section 21.8.9
24h	SPIPC4	SPI Pin Control Register 4	Section 21.8.10
28h	SPIPC5	SPI Pin Control Register 5	Section 21.8.11
2Ch	SPIPC6	SPI Pin Control Register 6	Section 21.8.12
30h	SPIPC7	SPI Pin Control Register 7	Section 21.8.13
34h	SPIPC8	SPI Pin Control Register 8	Section 21.8.14
38h	SPIDAT0	SPI Transmit Data Register 0	Section 21.8.15
3Ch	SPIDAT1	SPI Transmit Data Register 1	Section 21.8.16
40h	SPIBUF	SPI Receive Buffer Register	Section 21.8.17
44h	SPIEMU	SPI Emulation Register	Section 21.8.18
48h	SPIDELAY	SPI Delay Register	Section 21.8.19
4Ch	SPIDEF	SPI Default Chip Select Register	Section 21.8.20
50h-5Ch	SPIFMT	SPI Data Format Registers	Section 21.8.21
60h	INTVECT0	Interrupt Vector 0	Section 21.8.22
64h	INTVECT1	Interrupt Vector 1	Section 21.8.23
68h	SPIPC9[(1)]	SPI Pin Control Register 9	Section 21.8.24
70h	MIBSPIE	Multi-buffer Mode Enable Register	Section 21.8.25
74h	TGITENST	TG Interrupt Enable Set Register	Section 21.8.26
78h	TGITENCR	TG Interrupt Enable Clear Register	Section 21.8.27
7Ch	TGITLVST	Transfer Group Interrupt Level Set Register	Section 21.8.28
80h	TGITLVCR	Transfer Group Interrupt Level Clear Register	Section 21.8.29
84h	TGINTFLG	Transfer Group Interrupt Flag Register	Section 21.8.30
90h	TICKCNT	Tick Count Register	Section 21.8.31
94h	LTGPEND	Last TG End Pointer	Section 21.8.32
98h-D4h	TGxCTRL	TGx Control Registers	Section 21.8.33
120h	UERRCTRL	Multi-buffer RAM Uncorrectable Parity Error Control Register	Section 21.8.34
124h	UERRSTAT	Multi-buffer RAM Uncorrectable Parity Error Status Register	Section 21.8.35
128h	UERRADDR1	RXRAM Uncorrectable Parity Error Address Register	Section 21.8.36

(1) SPIPC9 only applies to SPI2.

Table 21-3. SPI Registers (continued)

Offset	Acronym	Register Description	Section
12Ch	UERRADR0	TXRAM Uncorrectable Parity Error Address Register	Section 21.8.37
130h	RXOVRN_BUF_ADDR	RXRAM Overrun Buffer Address Register	Section 21.8.38
134h	IOLPBKTSTCR	I/O Loopback Test Control Register	Section 21.8.39
138h	EXTENDED_PRESCALE1	SPI Extended Prescale Register 1	Section 21.8.40
13Ch	EXTENDED_PRESCALE2	SPI Extended Prescale Register 2	Section 21.8.41

21.8.1 SPI Global Control Register 0 (SPIGCR0)

Figure 21-17. SPI Global Control Register 0 (SPIGCR0) [offset = 00]

31	16
Reserved	
R-0	

15	1	0
Reserved		nRESET
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 21-4. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	nRESET		This is the local reset control for the module. This bit needs to be set to 1 before any operation on SPI / MibSPI can be done. Only after setting this bit to 1, the Auto Initialization of Multi-buffer RAM starts. Clearing this bit to 0 results in all of the control and status register values to return to their default values.
		0	SPI is in the reset state.
		1	SPI is out of the reset state.

21.8.2 SPI Global Control Register 1 (SPIGCR1)

Figure 21-18. SPI Global Control Register 1 (SPIGCR1) [offset = 04h]

31	25	24	23	17	16
Reserved		SPIEN	Reserved		LOOPBACK
R-0		R/W-0	R-0		R/WP-0

15	9	8	7	2	1	0
Reserved		POWERDOWN	Reserved		CLKMOD	MASTER
R-0		R/W-0	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 21-5. SPI Global Control Register 1 (SPIGCR1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	SPIEN		SPI enable. This bit enables SPI transfers. This bit must be set to 1 after all other SPI configuration bits have been written. When the SPIEN bit is 0 or cleared to 0, the following SPI registers get forced to their default states: • Both TX and RX shift registers • The TXDATA fields of the SPI Transmit Data Register 0 (SPIDAT0) and the SPI Transmit Data Register 1 (SPIDAT1) • All the fields of the SPI Flag Register (SPIFLG) • Contents of SPIBUF and the internal RXBUF registers
		0	The SPI is not activated for transfers.
		1	Activates SPI.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	LOOPBACK		Internal loop-back test mode. The internal self-test option can be enabled by setting this bit. If the SPISIMO and SPISOMI pins are configured with SPI functionality, then the SPISIMO[7:0] pins are internally connected to the SPISOMI[7:0] pins (transmit data is looped back as receive data). GIO mode for these pins is not supported in loopback mode. Externally, during loop-back operation, the SPICLK pin outputs an inactive value and SPISOMI[7:0] remains in the high-impedance state. If the SPI is initialized in slave mode or a data transfer is ongoing, errors may result. **Note: This loopback mode can only be used in master mode. Master mode must be selected before setting LOOPBACK. When this mode is selected, the CLKMOD bit should be set to 1, meaning that SPICLK is internally generated.**
		0	Internal loop-back test mode is disabled.
		1	Internal loop-back test mode is enabled.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	POWERDOWN		When active, the SPI state machine enters a power-down state.
		0	The SPI is in active mode.
		1	The SPI is in power-down mode.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	CLKMOD		Clock mode. This bit selects either an internal or external clock source. This bit also determines the I/O direction of the $\overline{\text{SPIENA}}$ and $\overline{\text{SPICS}}$ pins in functional mode.
		0	Clock is external. • $\overline{\text{SPIENA}}$ is an output. • $\overline{\text{SPICS}}$ are inputs.
		1	Clock is internally-generated. • $\overline{\text{SPIENA}}$ is an input. • $\overline{\text{SPICS}}$ are outputs.

Table 21-5. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
0	MASTER		SPISIMO/SPISOMI pin direction determination. Sets the direction of the SPISIMO and SPISOMI pins. **Note: For master-mode operation of the SPI, MASTER bit should be set to 1 and CLKMOD bit can be set either 1 or 0. The master-mode SPI can run on an external clock on SPICLK.** **For slave mode operation, both the MASTER and CLKMOD bits should be cleared to 0. Any other combinations may result in unpredictable behavior of the SPI. In slave mode. SPICLK will not be generated internally in slave mode.**
		0	SPISIMO[7:0] pins are inputs, SPISOMI[7:0] pins are outputs.
		1	SPISOMI[7:0] pins are inputs, SPISIMO[7:0] pins are outputs.

21.8.3 SPI Interrupt Register (SPIINT0)

Figure 21-19. SPI Interrupt Register (SPIINT0) [offset = 08h]

31						25	24
Reserved							ENABLEHIGHZ
R-0							R/W-0
23						17	16
Reserved							Reserved
R-0							R/W-0
15						10	9	8
Reserved							TXINT ENA	RXINT ENA
R-0							R/W-0	R/W-0
7	6	5	4	3	2	1	0
Reserved	RXOVRNINT ENA	Reserved	BITERR ENA	DESYNC ENA	PARERR ENA	TIMEOUT ENA	DLENERR ENA
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-6. SPI Interrupt Register (SPIINT0) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	ENABLEHIGHZ		$\overline{\text{SPIENA}}$ pin high-impedance enable. When active, the $\overline{\text{SPIENA}}$ pin (when it is configured as a WAIT functional output signal in a slave SPI) is forced to high-impedance when not driving a low signal. If inactive, then the pin will output both a high and a low signal.
		0	$\overline{\text{SPIENA}}$ pin is pulled high when not active.
		1	$\overline{\text{SPIENA}}$ pin remains high-impedance when not active.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	Reserved	0	Reserved. Do not use.
15-10	Reserved	0	Reads return 0. Writes have no effect.

Table 21-6. SPI Interrupt Register (SPIINT0) Field Descriptions (continued)

Bit	Field	Value	Description
9	TXINTENA		Causes an interrupt to be generated every time data is written to the shift register, so that the next word can be written to TXBUF. Setting this bit will generate an interrupt if the TXINTFLG bit (SPI Flag Register (SPIFLG)[9]) is set to 1.
		0	No interrupt will be generated upon TXINTFLG being set to 1.
		1	An interrupt will be generated upon TXINTFLG being set to 1.
			The transmitter empty interrupt is valid in compatibility mode of SPI only. In multi-buffered mode, this interrupts will not be generated, even if it is enabled.
			Note: An interrupt request will be generated as soon as this bit is set to 1. By default it will be generated on the INT0 line. The SPILVL register can be programmed to change the interrupt line.
8	RXINTENA		Causes an interrupt to be generated when the RXINTFLAG bit (SPI Flag Register (SPIFLG)[8]) is set by hardware.
		0	Interrupt will not be generated.
		1	Interrupt will be generated.
			The receiver full interrupt is valid in compatibility mode of SPI only. In multi-buffered mode, this interrupts will not be generated, even if it is enabled.
7	Reserved	0	Reads return 0. Writes have no effect.
6	RXOVRNINTENA		Overrun interrupt enable.
		0	Overrun interrupt will not be generated.
		1	Overrun interrupt will be generated.
5	Reserved	0	Reads return 0. Writes have no effect.
4	BITERRENA		Enables interrupt on bit error.
		0	No interrupt asserted upon bit error.
		1	Enables interrupt on bit error.
3	DESYNCENA		Enables interrupt on desynchronized slave. DESYNCENA is used in master mode only.
		0	No interrupt asserted upon desynchronization error.
		1	An interrupt is asserted on desynchronization of the slave (DESYNC = 1).
2	PARERRENA		Enables interrupt-on-parity-error.
		0	No interrupt asserted on parity error.
		1	An interrupt is asserted on a parity error.
1	TIMEOUTENA		Enables interrupt on ENA signal time-out.
		0	No interrupt asserted upon ENA signal time-out.
		1	An interrupt is asserted on a time-out of the ENA signal.
0	DLENERRENA		Data length error interrupt enable. A data length error occurs under the following conditions.
			Master: When $\overline{\text{SPIENA}}$ is used, if the $\overline{\text{SPIENA}}$ pin from the slave is deasserted before the master has completed its transfer, the data length error is set. That is, if the character length counter has not overflowed while $\overline{\text{SPIENA}}$ deassertion is detected, then it means that the slave has neither received full data from the master nor has it transmitted complete data.
			Slave: When $\overline{\text{SPICS}}$ pins are used, if the incoming valid $\overline{\text{SPICS}}$ pin is deactivated before the character length counter overflows, then the data length error is set.
		0	No interrupt is generated upon data length error.
		1	An interrupt is asserted when a data-length error occurs.

21.8.4 SPI Interrupt Level Register (SPILVL)

Figure 21-20. SPI Interrupt Level Register (SPILVL) [offset = 0Ch]

31							16
Reserved							
R-0							
15					10	9	8
Reserved						TXINT LVL	RXINT LVL
R-0						R/W-0	R/W-0
7	6	5	4	3	2	1	0
Reserved	RXOVRNINT LVL	Reserved	BITERR LVL	DESYNC LVL	PARERR LVL	TIMEOUT LVL	DLENERR LVL
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-7. SPI Interrupt Level Register (SPILVL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	TXINTLVL		Transmit interrupt level.
		0	Transmit interrupt is mapped to interrupt line INT0.
		1	Transmit interrupt is mapped to interrupt line INT1.
8	RXINTLVL		Receive interrupt level.
		0	Receive interrupt is mapped to interrupt line INT0.
		1	Receive interrupt is mapped to interrupt line INT1.
7	Reserved	0	Reads return 0. Writes have no effect.
6	RXOVRNINTLVL		Receive overrun interrupt level.
		0	Receive overrun interrupt is mapped to interrupt line INT0.
		1	Receive overrun interrupt is mapped to interrupt line INT1.
5	Reserved	0	Reads return 0. Writes have no effect.
4	BITERRLVL		Bit error interrupt level.
		0	Bit error interrupt is mapped to interrupt line INT0.
		1	Bit error interrupt is mapped to interrupt line INT1.
3	DESYNCLVL		Desynchronized slave interrupt level. (master mode only).
		0	An interrupt caused by desynchronization of the slave is mapped to interrupt line INT0.
		1	An interrupt caused by desynchronization of the slave is mapped to interrupt line INT1.
2	PARERRLVL		Parity error interrupt level.
		0	A parity error interrupt is mapped to interrupt line INT0.
		1	A parity error interrupt is mapped to interrupt line INT1.
1	TIMEOUTLVL		$\overline{\text{SPIENA}}$ pin time-out interrupt level.
		0	An interrupt on a time-out of the ENA signal (TIMEOUT = 1) is mapped to interrupt line INT0.
		1	An interrupt on a time-out of the ENA signal (TIMEOUT = 1) is mapped to interrupt line INT1.
0	DLENERRLVL		Data length error interrupt level (line) select.
		0	An interrupt on data length error is mapped to interrupt line INT0.
		1	An interrupt on data length error is mapped to interrupt line INT1.

21.8.5 SPI Flag Register (SPIFLG)

Software must check all flag bits when reading this register.

Figure 21-21. SPI Flag Register (SPIFLG) [offset = 10h]

31	25	24	23	16
Reserved		BUFINIT ACTIVE	Reserved	
R-0		R-0	R-0	

15	10	9	8
Reserved		TXINT FLG	RXINT FLG
R-0		R-0	R/W1C-0

7	6	5	4	3	2	1	0
Reserved	RXOVRNINT FLG	Reserved	BITERR FLG	DESYNC FLG	PARERR FLG	TIMEOUT FLG	DLENERR FLG
R-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 21-8. SPI Flag Register (SPIFLG) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUFINITACTIVE		Indicates the status of multi-buffer initialization process. Software can poll for this bit to determine if it can proceed with the register configuration of multi-buffer mode registers or buffer handling. **Note: If the SPIFLG register is read while the multi-buffer RAM is being initialized, the BUFINITACTIVE bit will be read as 1. If SPIFLG is read after the internal automatic buffer initialization is complete, this bit will be read as 0. This bit will show a value of 1 as long as the nRESET bit is 0, but does not really indicate that buffer initialization is underway. Buffer initialization starts only when the nRESET bit is set to 1.**
		0	Multi-buffer RAM initialization is complete.
		1	Multi-buffer RAM is still being initialized. Do not attempt to write to either multi-buffer RAM or any multi-buffer mode registers.
23-10	Reserved	0	Reads return 0. Writes have no effect.
9	TXINTFLG		Transmitter-empty interrupt flag. Serves as an interrupt flag indicating that the transmit buffer (TXBUF) is empty and a new word can be written to it. This flag is set when a word is copied to the shift register either directly from SPIDAT0/SPIDAT1 or from the TXBUF register. This bit is cleared by one of following methods: • Writing a new data to either SPIDAT0 or SPIDAT1 • Writing a 0 to SPIEN (SPIGCR1[24])
		0	Transmit buffer is now full. No interrupt pending for transmitter empty.
		1	Transmit buffer is empty. An interrupt is pending to fill the transmitter.

Table 21-8. SPI Flag Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
8	RXINTFLG		Receiver-full interrupt flag. This flag is set when a word is received and copied into the buffer register (SPIBUF). If RXINTEN is enabled, an interrupt is also generated. This bit is cleared under the following methods: • Reading the SPIBUF register • Reading TGINTVECT0 or TGINTVECT1 register when there is a receive buffer full interrupt • Writing a 1 to this bit • Writing a 0 to SPIEN (SPIGCR1[24]) • System reset During emulation mode, however, a read to the emulation register (SPIEMU) does not clear this flag bit.
		0	No new received data pending. Receive buffer is empty.
		1	A newly received data is ready to be read. Receive buffer is full.
			Note: Clearing RXINTFLG bit by writing a 1 before reading the SPIBUF sets the RXEMPTY bit of the SPIBUF register too. In this way, you can ignore a received word. However, if the internal RXBUF is already full, the data from RXBUF will be copied to SPIBUF and the RXEMPTY bit will be cleared again. The SPIBUF contents should be read first if this situation needs to be avoided.
7	Reserved	0	Reads return 0. Writes have no effect.
6	RXOVRNINTFLG		Receiver overrun flag. The SPI hardware sets this bit when a receive operation completes before the previous character has been read from the receive buffer. The bit indicates that the last received character has been overwritten and therefore lost. The SPI will generate an interrupt request if this bit is set and the RXOVRN INTEN bit (SPIINT0.6) is set high. This bit is cleared under the following conditions in compatibility mode of MibSPI: • Reading TGINTVECT0 or TGINTVECT1 register when there is a receive-buffer-overrun interrupt • Writing a 1 to RXOVRNINTFLG in the SPI Flag Register (SPIFLG) itself • Writing a 0 to SPIEN • Reading the data field of the SPIBUF register
			Note: Reading the SPIBUF register does not clear this RXOVRNINTFLG bit. If an RXOVRN interrupt is detected, then the SPIBUF may need to be read twice to get to the overrun buffer. This is due to the fact that the overrun will always occur to the internal RXBUF. Each read to the SPIBUF will result in RXBUF contents (if it is full) getting copied to SPIBUF.
			Note: There is a special condition under which the RXOVRNINTFLG flag gets set. If both SPIBUF and RXBUF are already full and while another reception is underway, if any errors (TIMEOUT, BITERR, and DLEN_ERR) occur, then RXOVR in RXBUF and RXOVRNINTFLG in SPIFLG registers will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a receive overrun.
			In multi-buffer mode of MibSPI, this bit is cleared under the following conditions: • Reading the RXOVRN_BUF_ADDR register • Writing a 1 to RXOVRNINTFLG in the SPI Flag Register (SPIFLG) itself In multi-buffer mode, if RXOVRNINTFLG is set, then the address of the buffer which experienced the overrun is available in RXOVRN_BUF_ADDR.
		0	Overrun condition did not occur.
		1	Overrun condition has occurred.
5	Reserved	0	Reads return 0. Writes have no effect.
4	BITERRFLG		Mismatch of internal transmit data and transmitted data. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIENA bit to 0
		0	No bit error occurred.
		1	A bit error occurred. The SPI samples the signal of the transmit pin (master: SIMO, slave: SOMI) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value a bit error is detected and the flag BITERRFLG is set. If BITERRENA is set an interrupt is asserted. Possible reasons for a bit error can be an excessively high bit rate, capacitive load, or another master/slave trying to transmit at the same time.

Table 21-8. SPI Flag Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
3	DESYNCFLG		Desynchronization of slave device. Desynchronization monitor is active in master mode only. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIENA bit to 0
		0	No slave desynchronization detected.
		1	A slave device is desynchronized. The master monitors the ENAble signal coming from the slave device and sets the DESYNC flag after the last bit is transmitted plus $t_{T2EDELAY}$. If DESYNCENA is set an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.
2	PARERRFLG		Calculated parity differs from received parity bit. If the parity generator is enabled (can be selected individually for each buffer) an even or odd parity bit is added at the end of a data word. During reception of the data word the parity generator calculates the reference parity and compares it to the received parity bit. In the event of a mismatch the PARITYERR flag is set and an interrupt is asserted if PARERRENA is set. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIENA bit to 0
		0	No parity error detected.
		1	A parity error occurred.
1	TIMEOUTFLG		Time-out caused by nonactivation of ENA signal. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIENA bit to 0
		0	No ENA-signal time-out occurred.
		1	An ENA signal time-out occurred. The SPI generates a time-out because the slave has not responded in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition the TIMEOUT flag in the status field of the corresponding buffer is set. The transmit request of the concerned buffer is cleared, that is, the SPI does not re-start a data transfer from this buffer.
0	DLENERRFLG		Data-length error flag. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIENA bit to 0 **Note: Whenever any transmission errors (TIMEOUT, BITERR, DLEN_ERR, PARITY_ERR, DESYNC) are detected and the error flags are cleared by writing to the error bit in the SPIFLG register, the corresponding error flag in SPIBUF does not get cleared. Software needs to read the SPIBUF until it becomes empty before proceeding. This ensures that all of the old status bits in SPIBUF are cleared before starting the next transfer.**
		0	No data length error has occurred.
		1	A data length error has occurred.

21.8.6 SPI Pin Control Register 0 (SPIPC0)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of SPIPC0 to SPIPC9 reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 21-22. SPI Pin Control Register 0 (SPIPC0) [offset = 14h]

31	24	23	16
Reserved		Reserved	
R/W-0		R/W-0	

15	12	11	10	9	8
Reserved		SOMIFUN0	SIMOFUN0	CLKFUN	ENAFUN
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	0
SCSFUN	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-9. SPI Pin Control (SPIPC0) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved. Do not use. **Note: Bit 24 is not physically implemented. It is a mirror of bit 11. Any write to bit 24 will be reflected on bit 11. When bit 24 and bit 11 are simultaneously written, the value of bit 11 will control the SPISOMI[0] pin. The read value of bit 24 always reflects the value of bit 11.**
23-16	Reserved	0	Reserved. Do not use. **Note: Bit 16 is not physically implemented. It is a mirror of bit 10. Any write to bit 16 will be reflected on bit 10. When bit 16 and bit 10 are simultaneously written, the value of bit 10 will control the SPISOMI[x] pin. The read value of bit 16 always reflects the value of bit 10.**
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIFUN0		Slave out, master in function. This bit determines whether the SPISOMI0 pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	The SPISOMI0 pin is a GIO pin.
		1	The SPISOMI0 pin is a SPI functional pin.
			Note: SPISOMI0 pin will always have to be programmed as functional pins for any SPI transfers.
10	SIMOFUN0		Slave in, master out function. This bits determine whether each SPISIMO0 pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	The SPISIMO0 pin is a GIO pin.
		1	The SPISIMO0 pin is a SPI functional pin.
			Note: SPISIMO0 pin will always have to be programmed as functional pins for any SPI transfers.
9	CLKFUN		SPI clock function. This bit determines whether the SPICLK pin is to be used as a general-purpose I/O pin, or as a SPI functional pin.
		0	The SPICLK pin is a GIO pin.
		1	The SPICLK pin is a SPI functional pin.
8	ENAFUN		$\overline{\text{SPIENA}}$ function. This bit determines whether the $\overline{\text{SPIENA}}$ pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	The $\overline{\text{SPIENA}}$ pin is a GIO pin.
		1	The $\overline{\text{SPIENA}}$ pin is a SPI functional pin.

Table 21-9. SPI Pin Control (SPIPC0) Field Descriptions (continued)

Bit	Field	Value	Description
7-0	SCSFUN		$\overline{\text{SPICS}}$ function. Determines whether each $\overline{\text{SPICS}}$ pin is to be used as a general-purpose I/O pin or as a SPI functional pin. If the slave $\overline{\text{SPICS}}$ pins are in functional mode and receive an inactive high signal, the slave SPI will place its output in a high-impedance state and disable shifting.
		0	The $\overline{\text{SPICS}}$ pin is a GIO pin.
		1	The $\overline{\text{SPICS}}$ pin is a SPI functional pin.

21.8.7 SPI Pin Control Register 1 (SPIPC1)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 21-23. SPI Pin Control Register 1 (SPIPC1) [offset = 18h]

31	24	23	16
SOMIDIR		SIMODIR	
R/W-0		R/W-0	

15	12	11	10	9	8
Reserved		SOMIDIR0	SIMODIR0	CLKDIR	ENADIR
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	0
SCSDIR	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-10. SPI Pin Control Register (SPIPC1) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIDIR		SPISOMIx direction. Controls the direction of each SPISOMIx pin when used for general-purpose I/O. If SPISOMIx pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register. **Note: Duplicate Control Bits for SPISOMI0. Bit 24 is not physically implemented. It is a mirror of bit 11. Any write to bit 24 will be reflected on bit 11. When bit 24 and bit 11 are simultaneously written, the value of bit 11 will control the SPISOMI pin. The read value of bit 24 always reflects the value of bit 11.**
		0	The SPISOMIx pin is an input.
		1	The SPISOMIx pin is an output.
23-16	SIMODIR		SPISIMOx direction. Controls the direction of each SPISIMOx pin when used for general-purpose I/O. If SPISIMOx pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register. **Note: Duplicate Control Bits for SPISIMO. Bit 16 is not physically implemented. It is a mirror of bit 10. Any write to bit 16 will be reflected on bit 10. When bit 16 and bit 10 are simultaneously written, the value of bit 10 will control the SPISIMO pin. The read value of bit 16 always reflects the value of bit 10.**
		0	The SPISIMOx pin is an input.
		1	The SPISIMOx pin is an output.
15-12	Reserved	0	Reads return 0. Writes have no effect.

Table 21-10. SPI Pin Control Register (SPIPC1) Field Descriptions (continued)

Bit	Field	Value	Description
11	SOMIDIR0		SPISOMI0 direction. This bit controls the direction of the SPISOMI0 pin when it is used as a general-purpose I/O pin. If the SPISOMI0 pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register.
		0	The SPISOMI0 pin is an input.
		1	The SPISOMI0 pin is an output.
10	SIMODIR0		SPISIMO0 direction. This bit controls the direction of the SPISIMO0 pin when it is used as a general-purpose I/O pin. If the SPISIMO0 pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register.
		0	The SPISIMO0 pin is an input.
		1	The SPISIMO0 pin is an output.
9	CLKDIR		SPICLK direction. This bit controls the direction of the SPICLK pin when it is used as a general-purpose I/O pin. In functional mode, the I/O direction is determined by the CLKMOD bit.
		0	The SPICLK pin is an input.
		1	The SPICLK pin is an output.
8	ENADIR		$\overline{\text{SPIENA}}$ direction. This bit controls the direction of the $\overline{\text{SPIENA}}$ pin when it is used as a general-purpose I/O. If the $\overline{\text{SPIENA}}$ pin is used as a functional pin, then the I/O direction is determined by the CLKMOD bit (SPIGCR1[1]).
		0	The $\overline{\text{SPIENA}}$ pin is an input.
		1	The $\overline{\text{SPIENA}}$ pin is an output.
7-0	SCSDIR		$\overline{\text{SPICS}}$ direction. These bits control the direction of each $\overline{\text{SPICS}}$ pin when it is used as a general-purpose I/O pin. Each pin could be configured independently from the others If the $\overline{\text{SPICS}}$ is used as a SPI functional pin; the I/O direction is determined by the CLKMOD bit (SPIGCR1[1]).
		0	The $\overline{\text{SPICS}}$ pin is an input.
		1	The $\overline{\text{SPICS}}$ pin is an output.

21.8.8 SPI Pin Control Register 2 (SPIPC2)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 21-24. SPI Pin Control Register 2 (SPIPC2) [offset = 1Ch]

31	24	23	16
SOMIDIN		SIMODIN	
R/W-U		R/W-U	

15	12	11	10	9	8
Reserved		SOMIDIN0	SIMODIN0	CLKDIN	ENADIN
R-0		R-U	R-U	R-U	R-U

7	0
SCSDIN	
R/W-U	

LEGEND: R/W = Read/Write; R = Read only; U = Undefined; -*n* = value after reset

Table 21-11. SPI Pin Control Register 2 (SPIPC2) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIDIN		SPISOMIx data in. The value of each SPISOMIx pin.
		0	The SPISOMIx pin is logic 0.
		1	The SPISOMIx pin is logic 1.
23-16	SIMODIN		SPISIMOx data in. The value of each SPISIMOx pin.
		0	The SPISIMOx pin is logic 0.
		1	The SPISIMOx pin is logic 1.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIDIN0		SPISOMI0 data in. The value of the SPISOMI0 pin.
		0	The SPISOMI0 pin is logic 0.
		1	The SPISOMI0 pin is logic 1.
10	SIMODIN0		SPISIMO0 data in. The value of the SPISIMO0 pin.
		0	The SPISIMO0 pin is logic 0.
		1	The SPISIMO0 pin is logic 1.
9	CLKDIN		Clock data in. The value of the SPICLK pin.
		0	The SPICLK pin is logic 0.
		1	The SPICLK pin is logic 1.
8	ENADIN		$\overline{\text{SPIENA}}$ data in. The value of the $\overline{\text{SPIENA}}$ pin.
		0	The $\overline{\text{SPIENA}}$ pin is logic 0.
		1	The $\overline{\text{SPIENA}}$ pin is logic 1.
7-0	SCSDIN		$\overline{\text{SPICS}}$ data in. The value of each $\overline{\text{SPICS}}$ pin.
		0	The $\overline{\text{SPICS}}$ pin is logic 0.
		1	The $\overline{\text{SPICS}}$ pin is logic 1.

21.8.9 SPI Pin Control Register 3 (SPIPC3)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 21-25. SPI Pin Control Register 3 (SPIPC3) [offset = 20h]

31	24	23	16
SOMIDOUT		SIMODOUT	
R/W-U		R/W-U	

15	12	11	10	9	8
Reserved		SOMIDOUT0	SIMODOUT0	CLKDOUT	ENADOUT
R-0		R/W-U	R/W-U	R/W-U	R/W-U

7	0
SCSDOUT	
R/W-U	

LEGEND: R/W = Read/Write; R = Read only; U = Undefined; -*n* = value after reset

Table 21-12. SPI Pin Control Register 3 (SPIPC3) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIDOUT		SPISOMIx data out write. This bit is only active when the SPISOMIx pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
			Bit 11 or bit 24 can be used to set the direction for pin SPISOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.
		0	Current value on SPISOMIx pin is logic 0.
		1	Current value on SPISOMIx pin is logic 1
23-16	SIMODOUT		SPISIMOx data out write. This bit is only active when the SPISIMOx pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
			Bit 10 or bit 16 can be used to set the direction for pin SPISIMO0. If a 32-bit write is performed, bit 10 will have priority over bit 16.
		0	Current value on SPISIMOx pin is logic 0.
		1	Current value on SPISIMOx pin is logic 1.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIDOUT0		SPISOMI0 data out write. This bit is only active when the SPISOMI0 pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	Current value on SPISOMI0 pin is logic 0.
		1	Current value on SPISOMI0 pin is logic 1.
10	SIMODOUT0		SPISIMO0 data out write. This bit is only active when the SPISIMO0 pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	The SPISIMO0 pin is logic 0.
		1	The SPISIMO0 pin is logic 1.
9	CLKDOUT		SPICLK data out write. This bit is only active when the SPICLK pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	The SPICLK pin is logic 0.
		1	The SPICLK pin is logic 1.

Table 21-12. SPI Pin Control Register 3 (SPIPC3) Field Descriptions (continued)

Bit	Field	Value	Description
8	ENADOUT		$\overline{\text{SPIENA}}$ data out write. Only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	The $\overline{\text{SPIENA}}$ pin is logic 0.
		1	The $\overline{\text{SPIENA}}$ pin is logic 1.
7-0	SCSDOUT		$\overline{\text{SPICS}}$ data out write. Only active when the $\overline{\text{SPICS}}$ pins are configured as a general-purpose I/O pins and configured as output pins. The value of these bits indicates the value sent to the pins.
		0	The $\overline{\text{SPICS}}$ pin is logic 0.
		1	The $\overline{\text{SPICS}}$ pin is logic 1.

21.8.10 SPI Pin Control Register 4 (SPIPC4)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 21-26. SPI Pin Control Register 4 (SPIPC4) [offset = 24h]

31–24	23–16
SOMISET	SIMOSET
R/W-U	R/W-U

15–12	11	10	9	8
Reserved	SOMISET0	SIMOSET0	CLKSET	ENASET
R-0	R/W-U	R/W-U	R/W-U	R/W-U

7–0
SCSSET
R/W-U

LEGEND: R/W = Read/Write; R = Read only; U = Undefined; -*n* = value after reset

Table 21-13. SPI Pin Control Register 4 (SPIPC4) Field Descriptions

Bit	Field	Value	Description
31-24	SOMISET		SPISOMIx data out set. This pin is only active when the SPISOMIx pin is configured as a general-purpose output pin.
			Bit 11 or bit 24 can be used to set the SPISOMI0 pin. If a 32-bit write is performed, bit 11 will have priority over bit 24.
		0	Read: SPISOMIx pin is logic 0.
			Write: Writing a 0 to this bit has no effect.
		1	Read: SPISOMIx pin is logic 1.
			Write: Logic 1 is placed on SPISOMIx pin, if it is in general-purpose output mode.
23-16	SIMOSET		SPISIMOx data out set. This bit is only active when the SPISIMOx pin is configured as a general-purpose output pin.
			Bit 10 or bit 16 can be used to set the SPISIMO0 pin. If a 32-bit write is performed, bit 10 will have priority over bit 16.
		0	Read: SPISIMOx pin is logic 0.
			Write: Writing a 0 to this bit has no effect.
		1	Read: SPISIMOx pin is logic 1.
			Write: Logic 1 is placed on SPISIMOx pin, if it is in general-purpose output mode.

Table 21-13. SPI Pin Control Register 4 (SPIPC4) Field Descriptions (continued)

Bit	Field	Value	Description
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMISET0		SPISOMI0 data out set. This pin is only active when the SPISOMI0 pin is configured as a general-purpose output pin.
		0	Read: SPISOMI0 pin is logic 0.
			Write: Writing a 0 to this bit has no effect.
		1	Read: SPISOMI0 pin is logic 1.
			Write: Logic 1 is placed on SPISOMI0 pin, if it is in general-purpose output mode.
10	SIMOSET0		SPISIMO0 data out set. This pin is only active when the SPISIMO0 pin is configured as a general-purpose output pin.
		0	Read: SPISIMO0 pin is logic 0.
			Write: Writing a 0 to this bit has no effect.
		1	Read: SPISIMO0 pin is logic 1.
			Write: Logic 1 is placed on SPISIMO0 pin, if it is in general-purpose output mode.
9	CLKSET		SPICLK data out set. This bit is only active when the SPICLK pin is configured as a general-purpose output pin.
		0	Read: SPICLK pin is logic 0.
			Write: Writing a 0 to this bit has no effect.
		1	Read: SPICLK pin is logic 1.
			Write: Logic 1 is placed on SPICLK pin, if it is in general-purpose output mode.
8	ENASET		$\overline{\text{SPIENA}}$ data out set. This bit is only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose output pin.
		0	Read: $\overline{\text{SPIENA}}$ pin is logic 0.
			Write: Writing a 0 to this bit has no effect.
		1	Read: $\overline{\text{SPIENA}}$ pin is logic 1.
			Write: Logic 1 is placed on $\overline{\text{SPIENA}}$ pin, if it is in general-purpose O/P mode.
7-0	SCSSET		$\overline{\text{SPICS}}$ data out set. This bit is only active when the $\overline{\text{SPICS}}$ pin is configured as a general-purpose output pin. A value of 1 written to this bit sets the corresponding SCSDOUT bit to 1.
		0	Read: $\overline{\text{SPICS}}$ pin is logic 0.
			Write: Writing a 0 to this bit has no effect.
		1	Read: $\overline{\text{SPICS}}$ pin is logic 1.
			Write: Logic 1 is placed on $\overline{\text{SPICS}}$ pin, if it is in general-purpose output mode.

21.8.11 SPI Pin Control Register 5 (SPIPC5)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 21-27. SPI Pin Control Register 5 (SPIPC5) [offset = 28h]

31	24	23	16
SOMICLR		SIMOCLR	
R/W-U		R/W-U	

15	12	11	10	9	8
Reserved		SOMICLR0	SIMOCLR0	CLKCLR	ENACLR
R-0		R/W-U	R/W-U	R/W-U	R/W-U

7	0
SCSCLR	
R/W-U	

LEGEND: R/W = Read/Write; R = Read only; U = Undefined; -*n* = value after reset

Table 21-14. SPI Pin Control Register 5 (SPIPC5) Field Descriptions

Bit	Field	Value	Description
31-24	SOMICLR		SPISOMIx data out clear. This pin is only active when the SPISOMIx pin is configured as a general-purpose output pin. **Bit 11 or bit 24 can be used to set the SPISOMI0 pin. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Read: The current value on SOMIDOUTx is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SOMIDOUTx is 1. Write: Logic 0 is placed on SPISOMIx pin if it is in general-purpose output mode.
23-16	SIMOCLR		SPISIMOx data out clear. This bit is only active when the SPISIMOx pin is configured as a general-purpose output pin. **Bit 10 or bit 16 can be used to set the SPISIMO0 pin. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Read: The current value on SOMODOUTx is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SOMODOUTx is 1. Write: Logic 0 is placed on SPISIMIx pin if it is in general-purpose output mode.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMICLR0		SPISOMI0 data out clear. This pin is only active when the SPISOMI0 pin is configured as a general-purpose output pin.
		0	Read: The current value on SPISOMI0 is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SPISOMI0 is 1. Write: Logic 0 is placed on SPISOMI0 pin, if it is in general-purpose output mode.
10	SIMOCLR0		SPISIMO0 data out clear. This pin is only active when the SPISIMO0 pin is configured as a general-purpose output pin.
		0	Read: The current value on SPISIMO0 is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SPISIMO0 is 1. Write: Logic 0 is placed on SPISIMO0 pin, if it is in general-purpose output mode.

Table 21-14. SPI Pin Control Register 5 (SPIPC5) Field Descriptions (continued)

Bit	Field	Value	Description
9	CLKCLR		SPICLK data out clear. This bit is only active when the SPICLK pin is configured as a general-purpose output pin.
		0	Read: The current value on SPICLK is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SPICLK is 1. Write: Logic 0 is placed on SPICLK pin, if it is in general-purpose output mode.
8	ENACLR		$\overline{\text{SPIENA}}$ data out clear. This bit is only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose output pin. A value of 1 written to this bit clears the corresponding ENABLEDOUT bit to 0.
		0	Read: The current value on $\overline{\text{SPIENA}}$ is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on $\overline{\text{SPIENA}}$ is 1. Write: Logic 0 is placed on $\overline{\text{SPIENA}}$ pin, if it is in general-purpose output mode.
7-0	SCSCLR		$\overline{\text{SPICS}}$ data out clear. This bit is only active when the $\overline{\text{SPICS}}$ pin is configured as a general-purpose output pin.
		0	Read: The current value on SCSDOUT is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SCSDOUT is 1. Write: Logic 0 is placed on $\overline{\text{SPICS}}$ pin, if it is in general-purpose output mode.

21.8.12 SPI Pin Control Register 6 (SPIPC6)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of SPIPC0 to SPIPC9 reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 21-28. SPI Pin Control Register 6 (SPIPC6) [offset = 2Ch]

31–24	23–16
SOMIPDR	SIMOPDR
R/W-0	R/W-0

15–12	11	10	9	8
Reserved	SOMIPDR0	SIMOPDR0	CLKPDR	ENAPDR
R-0	R/W-0	R/W-0	R/W-0	R/W-0

7–0
SCSPDR
R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-15. SPI Pin Control Register 6 (SPIPC6) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIPDR		SPISOMIx open drain enable. This bit enables open drain capability for each SPISOMIx pin, if the following conditions are met: • SOMIDIRx = 1 (SPISOMIx pin is configured in GIO mode as an output pin) • SOMIDOUTx = 1 **Bit 11 or bit 24 can both be used to enable open-drain for SPISOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Output value on the SPISOMIx pin is logic 1.
		1	Output pin SPISOMIx is in a high-impedance state.
23-16	SIMOPDR		SPISIMOx open drain enable. This bit enables open drain capability for each SPISIMOx pin, if the following conditions are met: • SIMODIRx = 1 (SPISIMOx pin is configured in GIO mode as an output pin) • SIMODOUTx = 1 **Bit 10 or bit 16 can both be used to enable open-drain for SPISIMO0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Output value on the SPISIMOx pin is logic 1.
		1	Output pin SPISIMOx is in a high-impedance state.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIPDR0		SPISOMI0 open-drain enable. This bit enables open-drain capability for SPISOMI0, if the following conditions are met. • SPISOMI0 pin is configured in GIO mode as an output pin • Output value on SPISOMI0 pin is logic 1.
		0	Output value on the SPISOMI0 pin is logic 1.
		1	Output pin SPISOMI0 is in a high-impedance state.
10	SIMOPDR0		SPISIMO0 open-drain enable. This bit enables open -drain capability for the SPISIMO0 pin, if the following conditions are met. • SPISIMO0 pin is configured in GIO mode as an output pin • Output value on SPISIMO0 pin is logic 1.
		0	Output value on the SPISIMO0 pin is logic 1.
		1	Output pin SPISIMO0 is in a high-impedance state.
9	CLKPDR		SPICLK open drain enable. This bit enables open drain capability for the SPICLK pin, if the following conditions are met: • SPICLK pin is configured in GIO mode as an output pin • SPICLKDOUT = 1
		0	Output value on the SPICLK pin is logic 1.
		1	Output pin SPICLK is in a high-impedance state.
8	ENAPDR		$\overline{\text{SPIENA}}$ pin open drain enable. This bit enables open drain capability for the $\overline{\text{SPIENA}}$ pin, if the following conditions are met: • $\overline{\text{SPIENA}}$ pin configured in GIO mode as an output pin • SPIENADOUT = 1
		0	Output value on the $\overline{\text{SPIENA}}$ pin is logic 1.
		1	Output pin $\overline{\text{SPIENA}}$ is in a high-impedance state.
7-0	SCSPDR		$\overline{\text{SPICS}}$ open drain enable. This bit enables open drain capability for each $\overline{\text{SPICS}}$ pin, if the following conditions are met: • $\overline{\text{SPICS}}$ pin is configured in GIO mode as an output pin • SCSDOUT = 1
		0	Output value on the $\overline{\text{SPICS}}$ pin is logic 1.
		1	Output pin $\overline{\text{SPICS}}$ is in a high-impedance state.

21.8.13 SPI Pin Control Register 7 (SPIPC7)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

NOTE: Default Register Value

The default values of these register bits vary by device. See your device datasheet for information about default pin states, which correspond to the register reset values (see the pin-list table).

Figure 21-29. SPI Pin Control Register 7 (SPIPC7) [offset = 30h]

31	24	23	16
SOMIDIS		SIMODIS	
R/W-n		R/W-n	

15	12	11	10	9	8
Reserved		SOMIPDIS0	SIMOPDIS0	CLKPDIS	ENAPDIS
R-0		R/W-n	R/W-n	R/W-n	R/W-n

7	0
SCSPDIS	
R/W-n	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-16. SPI Pin Control Register 7 (SPIPC7) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIDIS		SPISOMIx pull control disable. This bit disables pull control capability for each SPISOMIx pin if it is in input mode, regardless of whether it is in functional or GIO mode.
			Note: Bit 11 or bit 24 can be used to set pull-disable for SPISOMIO. If a 32-bit write is performed, bit 11 will have priority over bit 24.
		0	Pull control on the SPISOMIx pin is enabled.
		1	Pull control on the SPISOMIx pin is disabled.
23-16	SIMODIS		SPISIMOx pull control disable. This bit disables pull control capability for the SPISIMOx pin if it is in input mode, regardless of whether it is in functional or GIO mode.
			Note: Bit 10 or bit 16 can be used to set pull-disable for SPISIMO0. If a 32-bit write is performed, bit 10 will have priority over bit 16.
		0	Pull control on the SPISIMOx pin is enabled.
		1	Pull control on the SPISIMOx pin is disabled.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIPDIS0		SPISOMI0 pull control disable. This bit disables pull control capability for the pin SPISOMI0 pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the SPISOMI0 pin is enabled.
		1	Pull control on the SPISOMI0 pin is disabled.
10	SIMOPDIS0		SPISIMO0 pull control disable. This bit disables pull control capability for the pin SPISIMO0 pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the SPISIMO0 pin is enabled.
		1	Pull control on the SPISIMO0 pin is disabled.

Table 21-16. SPI Pin Control Register 7 (SPIPC7) Field Descriptions (continued)

Bit	Field	Value	Description
9	CLKPDIS		SPICLK pull control disable. This bit disables pull control capability for the pin SPICLK pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the SPICLK pin is enabled.
		1	Pull control on the SPICLK pin is disabled.
8	ENAPDIS		$\overline{\text{SPIENA}}$ pull control disable. This bit disables pull control capability for the pin $\overline{\text{SPIENA}}$ pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the $\overline{\text{SPIENA}}$ pin is enabled.
		1	Pull control on the $\overline{\text{SPIENA}}$ pin is disabled.
7-0	SCSPDIS		$\overline{\text{SPICS}}$ pull control disable. This bit disables pull control capability for each $\overline{\text{SPICS}}$ pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the $\overline{\text{SPICS}}$ pin is enabled.
		1	Pull control on the $\overline{\text{SPICS}}$ pin is disabled.

21.8.14 SPI Pin Control Register 8 (SPIPC8)

NOTE: Register bits vary by device

Register bits 31-24 and 23-16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

NOTE: Default Register Value

The default values of these register bits vary by device. See your device datasheet for information about default pin states, which correspond to the register reset values (see the pin-list table).

Figure 21-30. SPI Pin Control Register 8 (SPIPC8) [offset = 34h]

31–24	23–16
SOMIPSEL	SIMOPSEL
R/W-n	R/W-n

15–12	11	10	9	8
Reserved	SOMIPSEL0	SIMOPSEL0	CLKPSEL	ENAPSEL
R-0	R/W-n	R/W-n	R/W-n	R/W-n

7–0
SCSPSEL
R/W-n

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-17. SPI Pin Control Register 8 (SPIPC8) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIPSEL		SPISOMIx pull select. This bit selects the type of pull logic for each SPISOMIx pin.
			Note: Bit 11 or bit 24 can be used to set pull-select for SPISOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.
		0	Pull down on the SPISOMIx pin.
		1	Pull up on the SPISOMIx pin.

Table 21-17. SPI Pin Control Register 8 (SPIPC8) Field Descriptions (continued)

Bit	Field	Value	Description
23-16	SIMOPSEL		SPISIMOx pull select. This bit selects the type of pull logic for each SPISIMOx pin.
			Note: Bit 10 or bit 16 can be used to set pull-select for SPISIMO0. If a 32-bit write is performed, bit 10 will have priority over bit 16.
		0	Pull down on the SPISIMOx pin.
		1	Pull up on the SPISIMOx pin.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIPSEL0		SPISOMI0 pull select. This bit selects the type of pull logic at the SPISOMI0 pin.
		0	Pull down on the SPISOMI0 pin.
		1	Pull up on the SPISOMI0 pin.
10	SIMOPSEL0		SPISIMO0 pull select. This bit selects the type of pull logic at the SPISIMO0 pin.
		0	Pull down on the SPISIMO0 pin.
		1	Pull up on the SPISIMO0 pin.
9	CLKPSEL		SPICLK pull select. This bit selects the type of pull logic at the SPICLK pin.
		0	Pull down on the SPICLK pin.
		1	Pull up on the SPICLK pin.
8	ENAPSEL		$\overline{\text{SPIENA}}$ pull select. This bit selects the type of pull logic at the $\overline{\text{SPIENA}}$ pin.
		0	Pull down on the $\overline{\text{SPIENA}}$ pin.
		1	Pull up on the $\overline{\text{SPIENA}}$ pin.
7-0	SCSPSEL		$\overline{\text{SPICS}}$ pull select. This bit selects the type of pull logic for each $\overline{\text{SPICS}}$ pin.
		0	Pull down on the $\overline{\text{SPICS}}$ pin.
		1	Pull up on the $\overline{\text{SPICS}}$ pin.

21.8.15 SPI Transmit Data Register 0 (SPIDAT0)

Figure 21-31. SPI Transmit Data Register 0 (SPIDAT0) [offset = 38h]

31	16
Reserved	
R-0	

15	0
TXDATA	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-18. SPI Transmit Data Register 0 (SPIDAT0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	TXDATA	0-FFFFh	SPI transmit data. When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, TXBUF holds the written data. SPIEN (SPICGR1[24]) must be set to 1 before this register can be written to. Writing a 0 to the SPIEN register forces the lower 16 bits of the SPIDAT0 to 0x00.
			Note: When this register is read, the contents TXBUF, which holds the latest written data, will be returned.
			Note: Regardless of character length, the transmit word should be right-justified before writing to the SPIDAT1 register.
			Note: The default data format control register for SPIDAT0 is SPIFMT0. However, it is possible to reprogram the DFSEL[1:0] fields of SPIDAT1 before using SPIDAT0, to select a different SPIFMTx register.
			Note: It is highly recommended to use SPIDAT1 register, SPIDAT0 is supported for compatibility reasons.

21.8.16 SPI Transmit Data Register 1 (SPIDAT1)

NOTE: Writing to only the control fields, bits 28 through 16, does not initiate any SPI transfer in master mode. This feature can be used to set up SPICLK phase or polarity before actually starting the transfer by only updating the DFSEL bit field to select the required phase and polarity combination.

Figure 21-32. SPI Transmit Data Register 1 (SPIDAT1) [offset = 3Ch]

31 29	28	27	26	25 24	23 16
Reserved	CSHOLD	Rsvd	WDEL	DFSEL	CSNR
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

15 0
TXDATA
R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-19. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return 0. Writes have no effect.
28	CSHOLD		Chip select hold mode. The CSHOLD bit is supported in master mode only in compatibility-mode of SPI, (it is ignored in slave mode). CSHOLD defines the behavior of the chip select line at the end of a data transfer.
		0	The chip select signal is deactivated at the end of a transfer after the T2CDELAY time has passed. If two consecutive transfers are dedicated to the same chip select this chip select signal will be deactivated for at least 2VCLK cycles before it is activated again.
		1	The chip select signal is held active at the end of a transfer until a control field with new data and control information is loaded into SPIDAT1. If the new chip select number equals the previous one, the active chip select signal is extended until the end of transfer with CSHOLD cleared, or until the chip-select number changes.
27	Reserved	0	Reads return 0. Writes have no effect.
26	WDEL		Enable the delay counter at the end of the current transaction.
			Note: The WDEL bit is supported in master mode only. In slave mode, this bit will be ignored.
		0	No delay will be inserted. However, $\overline{\text{SPICS}}$ pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0.
			Note: The duration for which the $\overline{\text{SPICS}}$ pin remains deactivated depends upon the time taken to supply a new word after completing the shift operation. If TXBUF is already full, then the $\overline{\text{SPICS}}$ pin will be deasserted for at least two VCLK cycles (if WDEL = 0).
		1	After a transaction, WDELAY of the corresponding data format will be loaded into the delay counter. No transaction will be performed until the WDELAY counter overflows. The $\overline{\text{SPICS}}$ pins will be de-activated for at least (WDELAY + 2) × VCLK_Period duration.
25-24	DFSEL		Data word format select
		0	Data word format 0 is selected.
		1h	Data word format 1 is selected.
		2h	Data word format 2 is selected.
		3h	Data word format 3 is selected.
23-16	CSNR	0-FFh	Chip select (CS) number. CSNR defines the chip select pins that will be activated during the data transfer. CSNR is a bit-mask that controls all chip select pins. See Table 21-20.
			Note: If your MibSPI has less than 8 chip select pins, all unused upper bits will be 0. For example, the MiBSPI has 4 chip select pins, if you write FFh to CSNR, the actual number stored in CSNR is 1Fh.

Table 21-19. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions (continued)

Bit	Field	Value	Description
15-0	TXDATA	0-FFFFh	**Transfer data.** When written, these bits are copied to the shift register if it is empty. If the shift register is not empty, then they are held in TXBUF. SPIEN must be set to 1 before this register can be written to. Writing a 0 to SPIEN forces the lower 16 bits of SPIDAT1 to 0x0000. A write to this register (or to the TXDATA field only) drives the contents of the CSNR field on the $\overline{\text{SPICS}}$ pins, if the pins are configured as functional pins (automatic chip select, see Section 21.2). When this register is read, the contents of TXBUF, that holds the latest data written, will be returned. **Note: Regardless of the character length, the transmit data should be right-justified before writing to the SPIDAT1 register.**

Table 21-20. Chip Select Number Active

CSNR Value	Chip Select Active:			
	CS[3]	CS[2]	CS[1]	CS[0]
0h	No chip select pin is active.			
1h				x
2h			x	
3h			x	x
4h		x		
5h		x		x
6h		x	x	
7h		x	x	x
8h	x			
9h	x			x
Ah	x		x	
Bh	x		x	x
Ch	x	x		
Dh	x	x		x
Eh	x	x	x	
Fh	x	x	x	x
10h				
11h				x
12h			x	
13h			x	x
14h		x		
15h		x		x
16h		x	x	
17h		x	x	x
18h	x			
19h	x			x
1Ah	x		x	
1Bh	x		x	x
1Ch	x	x		
1Dh	x	x		x
1Eh	x	x	x	
1Fh	x	x	x	x

21.8.17 SPI Receive Buffer Register (SPIBUF)

Figure 21-33. SPI Receive Buffer Register (SPIBUF) [offset = 40h]

31	30	29	28	27	26	25	24
RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARITYERR	TIMEOUT	DLENERR
R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23							16
LCSNR							
R-0							
15							0
RXDATA							
R-0							

LEGEND: R = Read only; -*n* = value after reset

Table 21-21. SPI Receive Buffer Register (SPIBUF) Field Descriptions

Bit	Field	Value	Description
31	RXEMPTY		Receive data buffer empty. When the host reads the RXDATA field or the entire SPIBUF register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into RXDATA, and the RXEMPTY flag is cleared.
		0	New data has been received and copied into the RXDATA.
		1	No data has been received since the last read of RXDATA.
			This flag gets set to 1 under the following conditions: • Reading the RXDATA portion of the SPIBUF register • Writing a 1 to clear the RXINTFLG bit in the SPI Flag Register (SPIFLG) Write-clearing the RXINTFLG bit before reading the SPIBUF indicates the received data is being ignored. Conversely, RXINTFLG can be cleared by reading the RXDATA portion of SPIBUF (or the entire register).
30	RXOVR		Receive data buffer overrun. When a data transfer is completed and the received data is copied into RXBUF while it is already full, RXOVR is set. Overruns always occur to RXBUF, not to SPIBUF; the contents of SPIBUF are overwritten only after it is read by the Peripheral (VBUSP) master (CPU or other host processor). If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either SPI Flag Register (SPIFLG) or SPIVECTx shows the RXOVRN condition. Two read operations from the SPIBUF register are required to reach the overwritten buffer word (one to read SPIBUF, which then transfers RXDATA into SPIBUF for the second read). **Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.** **Note: A special condition under which RXOVR flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors such as TIMEOUT, BITERR and DLEN_ERR occur, then RXOVR in RXBUF and SPI Flag Register (SPIFLG) registers will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receive overrun.**
		0	No receive data overrun condition occurred since last read of the data field.
		1	A receive data overrun condition occurred since last read of the data field.
29	TXFULL		Transmit data buffer full. This flag is a read-only flag. Writing into the SPIDAT0 or SPIDAT1 field while the TX shift register is full will automatically set the TXFULL flag. Once the word is copied to the shift register, the TXFULL flag will be cleared. Writing to SPIDAT0 or SPIDAT1 when both TXBUF and the TX shift register are empty does not set the TXFULL flag.
		0	The transmit buffer is empty; SPIDAT0/SPIDAT1 is ready to accept a new data.
		1	The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data.

Table 21-21. SPI Receive Buffer Register (SPIBUF) Field Descriptions (continued)

Bit	Field	Value	Description
28	BITERR		Bit error. There was a mismatch of internal transmit data and transmitted data.
			Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.
		0	No bit error occurred.
		1	A bit error occurred. The SPI samples the signal of the transmit pins (master: SIMOx, slave: SOMIx) at the receive point (one-half clock cycle after the transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the BITERR flag is set. Possible reasons for a bit error include noise, an excessively high bit rate, capacitive load, or another master/slave trying to transmit at the same time.
27	DESYNC		Desynchronization of slave device. This bit is valid in master mode only.
			The master monitors the ENA signal coming from the slave device and sets the DESYNC flag if ENA is deactivated before the last reception point or after the last bit is transmitted plus $t_{T2EDELAY}$. If DESYNCENA is set, an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.
			Note: In the Compatibility Mode MibSPI, under some circumstances it is possible for a desync error detected for the previous buffer to be visible in the current buffer. This is because the receive completion flag/interrupt is generated when the buffer transfer is completed. But desynchronization is detected after the buffer transfer is completed. So, if the VBUS master reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition. In multi-buffer mode, the desync flag is always assured to be for the current buffer.
			Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.
		0	No slave desynchronization detected.
		1	A slave device is desynchronized.
26	PARITYERR		Parity error. The calculated parity differs from the received parity bit.
			If the parity generator is enabled (selected individually for each buffer) an even or odd parity bit is added at the end of a data word. During reception of the data word, the parity generator calculates the reference parity and compares it to the received parity bit. If a mismatch is detected, the PARITYERR flag is set.
			Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.
		0	No parity error detected.
		1	A parity error occurred.
25	TIMEOUT		Time-out because of non-activation of $\overline{\text{SPIENA}}$ pin.
			The SPI generates a time-out when the slave does not respond in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected, the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition, the TIMEOUT flag in the status field of the corresponding buffer and in the SPI Flag Register (SPIFLG) is set.
			Note: This bit is valid only in master mode.
			Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.
		0	No $\overline{\text{SPIENA}}$ pin time-out occurred.
		1	An $\overline{\text{SPIENA}}$ signal time-out occurred.
24	DLENERR		Data length error flag.
			Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.
		0	No data-length error has occurred.
		1	A data length error has occurred.
23-16	LCSNR	0-FFh	Last chip select number. LCSNR in the status field is a copy of CSNR in the corresponding control field. It contains the chip select number that was activated during the last word transfer.
15-0	RXDATA	0-FFFFh	SPI receive data. This is the received word, transferred from the receive shift-register at the end of a transfer. Regardless of the programmed character length and the direction of shifting, the received data is stored right-justified in the register.

21.8.18 SPI Emulation Register (SPIEMU)

Figure 21-34. SPI Emulation Register (SPIEMU) [offset = 44h]

31	16
Reserved	
R-8000h	
15	**0**
EMU_RXDATA	
R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 21-22. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	8000h	Reads return 0. Writes have no effect.
15-0	EMU_RXDATA	0-FFFFh	SPI receive data. The SPI emulation register is a mirror of the SPIBUF register. The only difference between SPIEMU and SPIBUF is that a read from SPIEMU does not clear any of the status flags.

21.8.19 SPI Delay Register (SPIDELAY)

Figure 21-35. SPI Delay Register (SPIDELAY) [offset = 48h]

31	24	23	16
C2TDELAY		T2CDELAY	
R/W-0		R/W-0	
15	**8**	**7**	**0**
T2EDELAY		C2EDELAY	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-23. SPI Delay Register (SPIDELAY) Field Descriptions

Bit	Field	Value	Description
31-24	C2TDELAY	0-FFh	Chip-select-active to transmit-start delay. See Figure 21-36 for an example. C2TDELAY is used only in master mode. It defines a setup time (for the slave device) that delays the data transmission from the chip select active edge by a multiple of VCLK cycles. The setup time value is calculated as follows. $t_{C2TDELAY}$ = (C2TDELAY + 2) × VCLK Period Example: VCLK = 25 MHz -> VCLK Period = 40ns; C2TDELAY = 07h; > $t_{C2TDELAY}$ = 360 ns When the chip select signal becomes active, the slave has to prepare data transfer within 360 ns. **Note: If phase = 1, the delay between SPICS falling edge to the first edge of SPICLK will have an additional 0.5 SPICLK period delay. This delay is as per the SPI protocol.**

Table 21-23. SPI Delay Register (SPIDELAY) Field Descriptions (continued)

Bit	Field	Value	Description
23-16	T2CDELAY	0-FFh	Transmit-end-to-chip-select-inactive-delay. See Figure 21-37 for an example. T2CDELAY is used only in master mode. It defines a hold time for the slave device that delays the chip select deactivation by a multiple of VCLK cycles after the last bit is transferred. The hold time value is calculated as follows: $t_{T2CDELAY}$ = (T2CDELAY +1) × VCLK Period Example: VCLK = 25 MHz -> VCLK Period = 40ns; T2CDELAY = 03h; > $t_{T2CDELAY}$ = 160 ns After the last data bit (or parity bit) is being transferred the chip select signal is held active for 160 ns. **Note: If phase = 0, then between the last edge of SPICLK and rise-edge of SPICS there will be an additional delay of 0.5 SPICLK period. This is as per the SPI protocol.** Both C2TDELAY and T2CDELAY counters do not have any dependency on the $\overline{\text{SPIENA}}$ pin value. Even if the $\overline{\text{SPIENA}}$ pin is asserted by the slave, the master will continue to delay the start of SPICLK until the C2TDELAY counter overflows. Similarly, even if the $\overline{\text{SPIENA}}$ pin is deasserted by the slave, the master will continue to hold the $\overline{\text{SPICS}}$ pins active until the T2CDELAY counter overflows. In this way, it is guaranteed that the setup and hold times of the $\overline{\text{SPICS}}$ pins are determined by the delay timers alone. To achieve better throughput, it should be ensured that these two timers are kept at the minimum possible values.
15-8	T2EDELAY	0-FFh	Transmit-data-finished to ENA-pin-inactive time-out. T2EDELAY is used in master mode only. It defines a time-out value as a multiple of SPI clock before $\overline{\text{SPIENA}}$ signal has to become inactive and after $\overline{\text{SPICS}}$ becomes inactive. SPICLK depends on which data format is selected. If the slave device is missing one or more clock edges, it becomes de-synchronized. In this case, although the master has finished the data transfer, the slave is still waiting for the missed clock pulses and the ENA signal is not disabled. The T2EDELAY defines a time-out value that triggers the DESYNC flag, if the $\overline{\text{SPIENA}}$ signal is not deactivated in time. The DESYNC flag is set to indicate that the slave device did not de-assert its $\overline{\text{SPIENA}}$ pin in time to acknowledge that it received all bits of the sent word. See Figure 21-38 for an example of this condition. **Note: DESYNC is also set if the SPI detects a de-assertion of $\overline{\text{SPIENA}}$ before the end of the transmission. The time-out value is calculated as follows:** $t_{T2EDELAY}$ = T2EDELAY/SPIclock Example: SPIclock = 8 Mbit/s; T2EDELAY = 10h; > $t_{T2EDELAY}$ = 2 µs The slave device has to disable the ENA signal within 2, otherwise DESYNC is set and an interrupt is asserted (if enabled).
7-0	C2EDELAY	0-FFh	Chip-select-active to ENA-signal-active time-out. C2EDELAY is used only in master mode and it applies only if the addressed slave generates an ENA signal as a hardware handshake response. C2EDELAY defines the maximum time between when the SPI activates the chip-select signal and the addressed slave has to respond by activating the ENA signal. C2EDELAY defines a time-out value as a multiple of SPI clocks. The SPI clock depends on whether data format 0 or data format 1 is selected. See Figure 21-39 for an example of this condition. **Note: If the slave device does not respond with the ENA signal before the time-out value is reached, the TIMEOUT flag in the SPIFLG register is set and a interrupt is asserted (if enabled).** If a time-out occurs, the SPI clears the transmit request of the timed-out buffer, sets the TIMEOUT flag for the current buffer, and continues with the transfer of the next buffer in the sequence that is enabled. The timeout value is calculated as follows: $t_{C2EDELAY}$ = C2EDELAY/SPIclock Example: SPIclock = 8 Mbit/s; C2EDELAY = 30 h; > $t_{C2EDELAY}$ = 6 ms The slave device has to activate the ENA signal within 6 ms after the SPI has activated the chip select signal ($\overline{\text{SPICS}}$), otherwise the TIMEOUT flag is set and an interrupt is asserted (if enabled).

Figure 21-36. Example: t $_{C2TDELAY}$= 8 VCLK Cycles

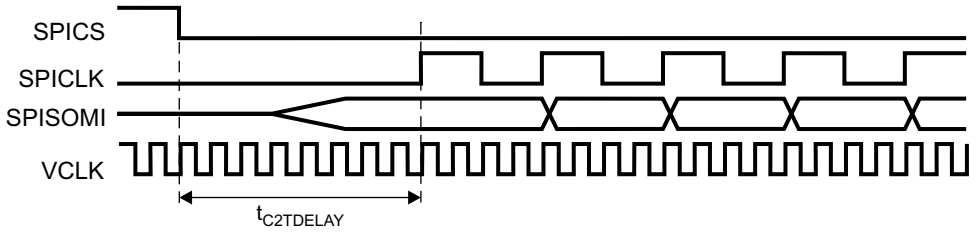


Figure 21-37. Example: t $_{T2CDELAY}$= 4 VCLK Cycles

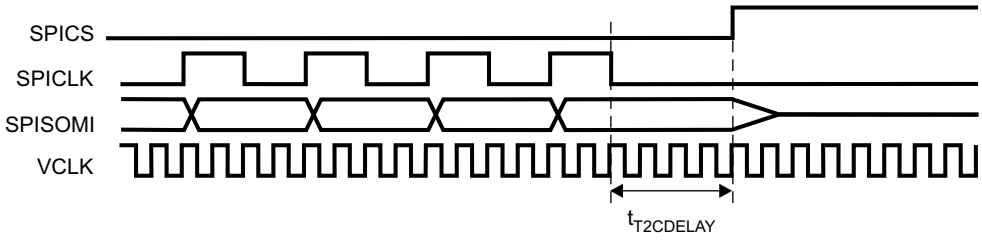


Figure 21-38. Transmit-Data-Finished-to-ENA-Inactive-Timeout

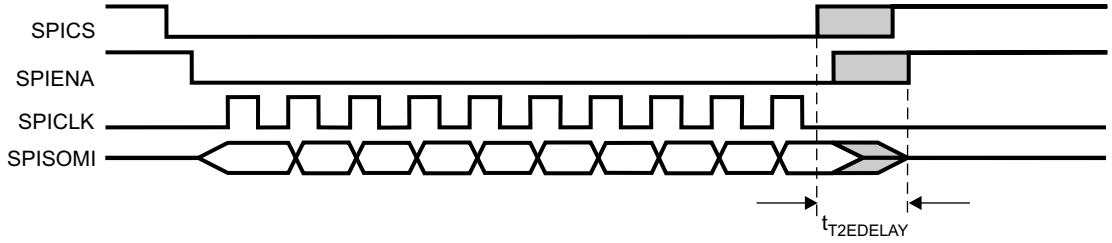
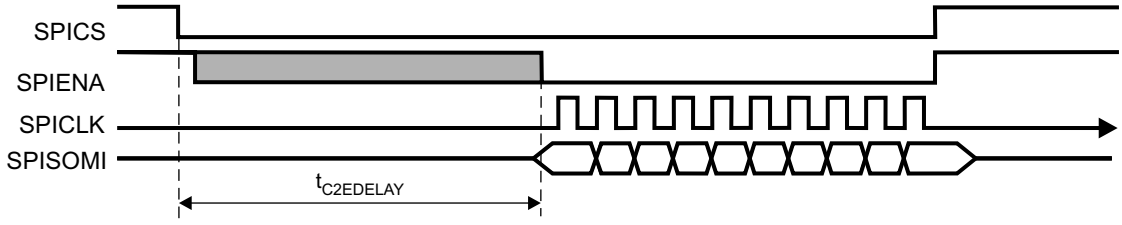


Figure 21-39. Chip-Select-Active-to-ENA-Signal-Active-Timeout



21.8.20 SPI Default Chip Select Register (SPIDEF)

Figure 21-40. SPI Default Chip Select Register (SPIDEF) [offset = 4Ch]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		CSDEF	
R-0		R/W-FFh	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-24. SPI Default Chip Select Register (SPIDEF) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	CDEF	0-FFh	Chip select default pattern. Master-mode only. The CSDEF bits are output to the $\overline{\text{SPICS}}$ pins when no transmission is being performed. It allows you to set a programmable chip-select pattern that deselects all of the SPI slaves.
		0	$\overline{\text{SPICS}}$ is cleared to 0 when no transfer is active.
		1	$\overline{\text{SPICS}}$ is set to 1 when no transfer is active.

21.8.21 SPI Data Format Registers (SPIFMT)

Figure 21-41. SPI Data Format Registers (SPIFMT[3:0]) [offset = 5Ch-50h]

31	30	29	28	27	26	25	24
WDELAY							
R/WP-0							

23	22	21	20	19	18	17	16
PARPOL	PARITYENA	WAITENA	SHIFTDIR	HDUPLEX_ ENAx	DIS_CS_ TIMERS	POLARITY	PHASE
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRESCALE								Reserved			CHARLEN				
R/WP-0								R-0			R/WP-0				

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; *-n* = value after reset

Table 21-25. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions

Bit	Field	Value	Description
31-24	WDELAY	0-FFh	Delay in between transmissions for data format x (x= 0,1,2,3).Idle time that will be applied at the end of the current transmission if the bit WDEL is set in the current buffer. The delay to be applied is equal to: WDELAY × P_{VCLK} + 2 × P_{VCLK} P_{VCLK} -> Period of VCLK.
23	PARPOL		Parity polarity: even or odd. PARPOLx can be modified in privilege mode only. It can be used for data format x (x= 0,1,2,3).
		0	An even parity flag is added at the end of the transmit data stream.
		1	An odd parity flag is added at the end of the transmit data stream.
22	PARITYENA		Parity enable for data format x. No parity generation/ verification is performed for this data format.
		0	A parity bit is transmitted at the end of each transmitted word. At the end of a transfer the parity generator compares the received parity bit with the locally-calculated parity flag. If the parity bits do not match the RXERR flag is set in the corresponding control field. The parity type (even or odd) can be selected via the PARPOL bit.
		1	**Note: If an uncorrectable error flag is set in a slave-mode SPI, then the wrong parity bit will be transmitted to indicate to the master that there has been some issue with the data parity. The SOMI pins will be forced to transmit all 0s, and the parity bit will be transmitted as 1 if even parity is selected and as 0 if odd parity is selected (using the PARPOLx bit of this register). This behavior occurs regardless of an uncorrectable parity error on either TXRAM or RXRAM.**
21	WAITENA		The master waits for the ENA signal from slave for data format x. WAITENA is valid in master mode only. WAITENA enables a flexible SPI network where slaves with ENA signal and slaves without ENA signal can be mixed. WAITENA defines, for each transferred word, whether the addressed slave generates the ENA signal or not.
		0	The SPI does not wait for the ENA signal from the slave and directly starts the transfer.
		1	Before the SPI starts the data transfer it waits for the ENA signal to become low. If the ENA signal is not pulled down by the addressed slave before the internal time-out counter (C2EDELAY) overflows, then the master aborts the transfer and sets the TIMEOUT error flag.
20	SHIFTDIR		Shift direction for data format x. With bit SHIFTDIRx, the shift direction for data format x (x=0,1,2,3) can be selected.
		0	MSB is shifted out first.
		1	LSB is shifted out first.

Table 21-25. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions (continued)

Bit	Field	Value	Description
19	HDUPLEX_ENAx		Half Duplex transfer mode enable for Data Format x. This bit controls the I/O function of SOMI/SIMO lines for a specific requirement where in the case of Master mode, TX pin - SIMO will act as an RX pin, and in the case of Slave mode, RX pin - SIMO will act as a TX pin..
		0	Normal Full Duplex transfer.
		1	If MASTER = 1, SIMO pin will act as an RX pin (No TX possible). If MASTER = 0, SIMO pin will act as a TX pin (No RX possible). For all normal operations, HDUPLEX_ENAx bits should always remain '0'. It is intended for the usage when the SIMO pin is used for both TX and RX operations at different times.
18	DIS_CS_TIMERS		Disable chip-select timers for this format.The C2TDELAY and T2CDELAY timers are by default enabled for all the data format registers. Using this bit, these timers can be disabled for a particular data format, if they are not required. When a master is handling multiple slaves, with varied set-up hold requirement, the application can selectively choose to include or not include the chip-select delay timers for any slaves.
		0	Both C2TDELAY and T2CDELAY counts are inserted for the chip selects.
		1	No C2TDELAY or T2CDELAY is inserted in the chip select timings.
17	POLARITY		SPI data format x clock polarity. POLARITYx defines the clock polarity of data format x. The following restrictions apply when switching clock phase and/or polarity: • In 3-pin/4-pin with nENA pin configuration of a slave SPI, the clock phase and polarity cannot be changed on-the-fly between two transfers. The slave should be reset and reconfigured if clock phase/polarity needs to be switched. In summary, SPI format switching is not fully supported in slave mode. • Even while using chip select pins, the polarity of SPICLK can be switched only while the slave is not selected by a valid chip select. The master SPI should ensure that while switching SPICLK polarity, it has deselected all of its slaves. Otherwise, the switching of SPICLK polarity may be incorrectly treated as a clock edge by some slaves.
		0	If POLARITYx is cleared to 0, the SPI clock signal is low-inactive, that is, before and after data transfer the clock signal is low.
		1	If POLARITYx is set to 1, the SPI clock signal is high-inactive, that is, before and after data transfer the clock signal is high.
16	PHASE		SPI data format x clock delay. PHASEx defines the clock delay of data format x.
		0	If PHASEx is cleared to 0, the SPI clock signal is not delayed versus the transmit/receive data stream. The first data bit is transmitted with the first clock edge and the first bit is received with the second (inverse) clock edge.
		1	If PHASEx is set to 1, the SPI clock signal is delayed by a half SPI clock cycle versus the transmit/receive data stream. The first transmit bit has to output prior to the first clock edge. The master and slave receive the first bit with the first edge.
15-8	PRESCALE		SPI data format x prescaler. PRESCALEx determines the bit transfer rate of data format x if the SPI is the network master. PRESCALEx is use to derive SPICLK from VCLK. If the SPI is configured as slave, PRESCALEx **does not need** to be configured. The clock rate for data format x can be calculated as: $BR_{Formatx}$ = VCLK / (PRESCALEx + 1) **Note: When PRESCALEx is cleared to 0, the SPI clock rate defaults to VCLK/2.**
7-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	CHARLEN	0-1Fh	SPI data format x data-word length. CHARLENx defines the word length of data format x. Legal values are 0x02 (data word length = 2 bit) to 10h (data word length = 16). Illegal values, such as 00 or 1Fh are not allowed; their effect is indeterminate.

21.8.22 Interrupt Vector 0 (INTVECT0)

NOTE: The TG interrupt is not available in MibSPI in compatibility mode. Therefore, there is no possibility to access this register in compatibility mode.

Figure 21-42. Interrupt Vector 0 (NTVECT0) [offset = 60h]

31	16
Reserved	
R-0	

15	6	5	1	0
Reserved		INTVECT0		SUSPEND0
R-0		R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 21-26. Transfer Group Interrupt Vector 0 (INTVECT0)

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-1	INTVECT0		INTVECT0. Interrupt vector for interrupt line INT0. Returns the vector of the pending interrupt at interrupt line INT0. If more than one interrupt is pending, INTVECT0 always references the highest prior interrupt source first. **Note:**This field reflects the status of the SPIFLG register in vector format. Any updates to the SPIFLG register will automatically cause updates to this field.
		0	There is no pending interrupt.
		1h + x	Transfer group x (x = 0 to 15) has a pending interrupt. SUSPEND0 reflects the type of interrupt (*suspended* or *finished*).
		11h	Error Interrupt pending. The lower half of SPIFLG contains more details about the type of error.
		13h	The pending interrupt is a Receive Buffer Overrun interrupt.
		12h	**SPI mode:** The pending interrupt is a Receive Buffer Full interrupt. **Mib mode:** Reserved. This bit combination should not occur.
		14h	**SPI mode:** The pending interrupt is a Transmit Buffer Empty interrupt. **Mib mode:** Reserved. This bit combination should not occur.
		All other values	**SPI mode:** Reserved. These bit combinations should not occur.
0	SUSPEND0		Transfer suspended / Transfer finished interrupt flag. Every time INTVECT0 is read by the host, the corresponding interrupt flag of the referenced transfer group is cleared and INTVECT0 is updated with the vector coming next in the priority chain.
		0	The interrupt type is a transfer finished interrupt. In other words, the buffer array referenced by INTVECT0 has asserted an interrupt because all of data from the transfer group has been transferred.
		1	The interrupt type is a "transfer suspended" interrupt. In other words, the transfer group referenced by INTVECT0 has asserted an interrupt because the buffer to be transferred next is in suspend-to-wait mode.

NOTE: Reading from the INTVECT0 register when Transmit Empty is indicated does not clear the TXINTFLG flag in the SPI Flag Register (SPIFLG). Writing a new word to the SPIDATx register clears the Transmit Empty interrupt.

NOTE: In multi-buffer mode, INTVECT0 contains the interrupt for the highest priority transfer group. A read from INTVECT0 automatically causes the next-highest priority transfer group's interrupt status to get loaded into INTVECT0 and its corresponding SUSPEND flag to get loaded into SUSPEND0. The transfer group with the lowest number has the highest priority, and the transfer group with the highest number has the lowest priority.

Reading the INTVECT0 register when the RXOVRN interrupt is indicated in multi-buffer mode does not clear the RXOVRN flag and hence does not clear the vector. The RXOVRN interrupt vector may be cleared in multi-buffer mode either by write-clearing the RXOVRN flag in the SPI Flag Register (SPIFLG) or by reading the RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR).

21.8.23 Interrupt Vector 1 (INTVECT1)

NOTE: The TG interrupt is not available in SPI in compatibility mode compatibility mode. Therefore, there is no possibility to access this register in compatibility mode.

Figure 21-43. Interrupt Vector 1 (INTVECT1) [offset = 64h]

31	16
Reserved	
R-0	

15	6	5	1	0
Reserved		INTVECT1		SUSPEND1
R-0		R-0		R-0

LEGEND: R = Read only; -*n* = value after reset

Table 21-27. Transfer Group Interrupt Vector 1 (INTVECT1)

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-1	INTVECT1		INTVECT1. Interrupt vector for interrupt line INT1.
			Returns the vector of the pending interrupt at interrupt line INT1. If more than one interrupt is pending, INTVECT1 always references the highest prior interrupt source first.
			Note: This field reflects the status of the SPIFLG register in vector format. Any updates to the SPIFLG register will automatically cause updates to this field.
		0	There is no pending interrupt. **SPI mode only.**
		11h	Error Interrupt pending. The lower half of SPIINT1 contains more details about the type of error. **SPI mode only.**
		13h	The pending interrupt is a Receive Buffer Overrun interrupt. **SPI mode only.**
		12h	The pending interrupt is a Receive Buffer Full interrupt. **SPI mode only.**
		14h	The pending interrupt is a Transmit Buffer Empty interrupt. **SPI mode only.**
		All other values	Reserved. These bit combinations should not occur. **SPI mode only.**
0	SUSPEND1		Transfer suspended / Transfer finished interrupt flag.
			Every time INTVECT1 is read by the host, the corresponding interrupt flag of the referenced transfer group is cleared and INTVECT1 is updated with the vector coming next in the priority chain.
		0	The interrupt type is a transfer finished interrupt. In other words, the buffer array referenced by INTVECT1 has asserted an interrupt because all of data from the transfer group has been transferred.
		1	The interrupt type is a transfer suspended interrupt. In other words, the transfer group referenced by INTVECT1 has asserted an interrupt because the buffer to be transferred next is in suspend-to-wait mode.

NOTE: Reading from the INTVECT1 register when Transmit Empty is indicated does not clear the TXINTFLG flag in the SPI Flag Register (SPIFLG). Writing a new word to the SPIDATx register clears the Transmit Empty interrupt.

NOTE: In multi-buffer mode, INTVECT1 contains the interrupt for the highest priority transfer group. A read from INTVECT1 automatically causes the next-highest priority transfer group's interrupt status to get loaded into INTVECT1 and its corresponding SUSPEND flag to get loaded into SUSPEND1. The transfer group with the lowest number has the highest priority, and the transfer group with the highest number has the lowest priority.

Reading the INTVECT1 register when the RXOVRN interrupt is indicated in multi-buffer mode does not clear the RXOVRN flag and hence does not clear the vector. The RXOVRN interrupt vector may be cleared in multi-buffer mode either by write-clearing the RXOVRN flag in the SPI Flag Register (SPIFLG) or by reading the RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR).

21.8.24 SPI Pin Control Register 9 (SPIPC9)

SPIPC9 only applies to SPI2.

Figure 21-44. SPI Pin Control Register 9 (SPIPC9) [offset = 68h]

31	25	24	23	17	16
Reserved		SOMISRS0	Reserved		SIMOSRS0
R-0		R/W-0	R-0		R/W-0

15	12	11	10	9	8	0
Reserved		SOMISRS0	SIMOSRS0	CLKSRS	Reserved	
R-0		R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-28. SPI Pin Control Register 9 (SPIPC9) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return the value that was last written. Writes have no effect.
24	SOMISRS0		SPI2 SOMI[0] slew control. This bit controls between the fast or slow slew mode.
			Note: Duplicate Control Bits for SPI2 SOMI[0]. Bit 24 is not physically implemented. It is a mirror of bit 11. Any write to bit 24 will be reflected on bit 11. When bit 24 and bit 11 are simultaneously written, the value of bit 11 will control the SPI2 SOMI[0] pin. The read value of bit 24 always reflects the value of bit 11.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
23-17	Reserved	0	Reads return the value that was last written. Writes have no effect.
16	SIMOSRS0		SPI2 SIMO[0] slew control. This bit controls between the fast or slow slew mode.
			Note: Duplicate Control Bits for SPI2 SIMO[0]. Bit 16 is not physically implemented. It is a mirror of bit 10. Any write to bit 16 will be reflected on bit 10. When bit 16 and bit 10 are simultaneously written, the value of bit 10 will control the SPI2 SIMO[0] pin. The read value of bit 16 always reflects the value of bit 10.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMISRS0		SPI2 SOMI[0] slew control. This bit controls between the fast or slow slew mode.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
10	SIMOSRS0		SPI2 SIMO[0] slew control. This bit controls between the fast or slow slew mode.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
9	CLKSRS		SPI2 CLK slew control. This bit controls between the fast or slow slew mode.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
8-0	Reserved	0	Reads return the value that was last written. Writes have no effect.

21.8.25 Multi-buffer Mode Enable Register (MIBSPIE)

NOTE: Accessibility of Multi-Buffer RAM

The multi-buffer RAM is not accessible unless the MSPIENA bit set to 1. The only exception to this is in test mode, where, by setting RXRAMACCESS to 1, the multi-buffer RAM can be fully accessed for both read and write.

Figure 21-45. Multi-buffer Mode Enable Register (MIBSPIE) [offset = 70h]

31	17	16
Reserved		RXRAM_ACCESS
R-0		R/WP-0

15	1	0
Reserved		MSPIENA
R-0		R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 21-29. Multi-buffer Mode Enable Register (MIBSPIE) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	RXRAM_ACCESS		Receive-RAM access control. During normal operating mode of SPI, the receive data/status portion of multi-buffer RAM is read-only. To enable testing of receive RAM, direct read/write access is enabled by setting this bit.
		0	The RX portion of multi-buffer RAM is not writable by the CPU.
		1	The whole of multi-buffer RAM is fully accessible for read/write by the CPU.
			Note: The RX RAM ACCESS bit remains 0 after reset and it should remain cleared to 0 at all times, except when testing the RAM. SPI should be given a local reset by using the nRESET (SPIGCR0[0]) bit after RAM testing is performed so that the multi-buffer RAM gets re-initialized.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	MSPIENA		Multi-buffer mode enable. After power-up or reset, MSPIENA remains cleared, which means that the SPI runs in compatibility mode by default. If multi-buffer mode is desired, this register should be configured first after configuring the SPIGCR0 register. If MSPIENA is not set to 1, the multi-buffer mode registers are not writable.
		0	The SPI runs in compatibility mode, that is, in this mode the MibSPI is fully code-compliant to the standard device SPI. No multi-buffered-mode features are supported.
		1	The SPI is configured to run in multi-buffer mode.

NOTE: Accessibility of Registers

Registers from this offset address onwards are not accessible in SPI compatibility mode. They are accessible only in the multi-buffer mode.

21.8.26 TG Interrupt Enable Set Register (TGITENST)

The register TGITENST contains the TG interrupt enable flags for transfer-finished and for transfer-suspended events. Each of the enable bits in the higher half-word and the lower half-word of TGITENST belongs to one TG.

The register map shown in Figure 21-46 and Table 21-30 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 21-46. TG Interrupt Enable Set Register (TGITENST) [offset = 74h]

31	16
SETINTENRDY[15:0]	
R/W-0	

15	0
SETINTENSUS[15:0]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-30. TG Interrupt Enable Set Register (TGITENST) Field Descriptions

Bit	Field	Value	Description
31-16	SETINTENRDY[*n*]		TG interrupt set (enable) when transfer finished. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx completes. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is enabled. The interrupt gets generated when TGx completes. Write: Enable the TGx-completed interrupt. The interrupt gets generated when TGx completes.
15-0	SETINTENSUS[*n*]		TG interrupt set (enabled) when transfer suspended. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx is suspended. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is enabled. The interrupt gets generated when TGx is suspended. Write: Enable the TGx-completed interrupt. The interrupt gets generated when TGx is suspended.

21.8.27 TG Interrupt Enable Clear Register (TGITENCR)

The register TGITENCR is used to clear the interrupt enables for the TG-completed interrupt and the TG-suspended interrupts.

The register map shown in Figure 21-47 and Table 21-31 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 21-47. TG Interrupt Enable Clear Register (TGITENCR) [offset = 78h]

31	16
CLRINTENRDY[15:0]	
R/W-0	
15	0
CLRINTENSUS[15:0]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-31. TG Interrupt Enable Clear Register (TGITENCR) Field Descriptions

Bit	Field	Value	Description
31-16	CLRINTENRDY[*n*]		TG interrupt clear (disabled) when transfer finished. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx completes. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is enabled. The interrupt gets generated when TGx completes. Write: Disable the TGx-completed interrupt. The interrupt does not get generated when TGx completes.
15-0	CLRINTENSUS[*n*]		TG interrupt clear (disabled) when transfer suspended. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx is suspended. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is enabled. The interrupt gets generated when TGx is suspended. Write: Disable the TGx-completed interrupt. The interrupt does not get generated when TGx is suspended.

21.8.28 Transfer Group Interrupt Level Set Register (TGITLVST)

The register TGITLVST sets the level of interrupts for transfer completed interrupt and for transfer suspended interrupt to level 1.

The register map shown in Figure 21-48 andTable 21-32 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 21-48. Transfer Group Interrupt Level Set Register (TGITLVST) [offset = 7Ch]

31	16
SETINTLVLRDY[15:0]	
R/W-0	

15	0
SETINTLVLSUS[15:0]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-32. Transfer Group Interrupt Level Set Register (TGITLVST) Field Descriptions

Bit	Field	Value	Description
31-16	SETINTLVLRDY[*n*]		Transfer-group completed interrupt level set. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is set to INT0. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is set to INT1. Write: Set the TGx-completed interrupt to INT1.
15-0	SETINTLVLSUS[*n*]		Transfer-group suspended interrupt level set. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on.
		0	Read: The TGx-suspended interrupt is set to INT0. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-suspended interrupt is set to INT1. Write: Set the TG-x suspended interrupt to INT1.

21.8.29 Transfer Group Interrupt Level Clear Register (TGITLVCR)

The register TGITLVCR clears the level of interrupts for transfer completed interrupt and for transfer suspended interrupt to level 0.

The register map shown in Figure 21-49 and Table 21-33 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 21-49. Transfer Group Interrupt Level Clear Register (TGITLVCR) [offset = 80h]

31	16
CLRINTLVLRDY[15:0]	
R/W-0	

15	0
CLRINTLVLSUS[15:0]	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-33. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions

Bit	Field	Value	Description
31-16	CLRINTLVLRDY[*n*]		Transfer-group completed interrupt level clear. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is set to INT0. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is set to INT1. Write: Clear the TGx-completed interrupt to INT0.
15-0	CLRINTLVLSUS[*n*]		Transfer group suspended interrupt level clear. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on.
		0	Read: The TGx-suspended interrupt is set to INT0. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-suspended interrupt is set to INT1. Write: Clear the TG-x suspended interrupt to INT0.

21.8.30 Transfer Group Interrupt Flag Register (TGINTFLAG)

The TGINTFLAG register comprises the transfer group interrupt flags for transfer-completed interrupts (INTFLGRDYx) and for transfer-suspended interrupts (INTFLGSUSx). Each of the interrupt flags in the higher half-word and the lower half-word of TGINTFLAG belongs to one TG.

The register map shown in Figure 21-50 and Table 21-34 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 21-50. Transfer Group Interrupt Flag Register (TGINTFLAG) [offset = 84h]

31	16
INTFLGRDY[15:0]	
R/W1C-0	

15	0
INTFLGSUS[15:0]	
R/W1C-0	

LEGEND: R/W = Read/Write; W1C = Write 1 to clear; -*n* = value after reset

Table 21-34. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions

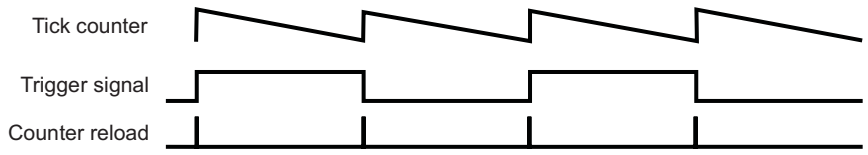
Bit	Field	Value	Description
31-16	INTFLGRDY[*n*]		Transfer-group interrupt flag for a transfer-completed interrupt. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on. **Note: Read Clear Behavior. Reading the interrupt vector registers TGINTVECT0 or TGINTVECT1 automatically clears the interrupt flag bit INTFLGRDYx referenced by the vector number given by INTVECT0/INTVECT1 bits, if the SUSPEND[0:1] bit in the vector registers is 0.**
		0	Read: No transfer-completed interrupt occurred since last clearing of the INTFLGRDYx flag. Write: A write of 0 to this bit has no effect.
		1	Read: A transfer finished interrupt from transfer group x occurred. No matter whether the interrupt is enabled or disabled (INTENRDYx = don't care) or whether the interrupt is mapped to INT0 or INT1, INTFLGRDYx is set right after the transfer from TGx is finished. Write: The corresponding bit flag is cleared.
15-0	INTFLGSUS[*n*]		Transfer-group interrupt flag for a transfer-suspend interrupt. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on. **Note: Read Clear Behavior. Reading the interrupt vector registers TGINTVECT0 or TGINTVECT1 automatically clears the interrupt flag bit INTFLGSUSx referenced by the vector number given by INTVECT0/INTVECT1 bits, if the SUSPEND[0:1] bit in the corresponding vector registers is 1.**
		0	Read: No transfer-suspended interrupt occurred since the last clearing of the INTFLGSUSx flag. Write: A write of 0 to this bit has no effect.
		1	Read: A transfer-suspended interrupt from TGx occurred. No matter whether the interrupt is enabled or disabled (INTENSUSx = don't care) or whether the interrupt is mapped to INT0 or INT1, INTFLGSUSx is set right after the transfer from transfer group x is suspended. Write: The corresponding bit flag is cleared.

21.8.31 Tick Count Register (TICKCNT)

One of the trigger sources for TGs is an internal periodic time trigger. This time trigger is called a tick counter and is basically a down-counter with a preload/reload value. Every time the tick counter detects an underflow it reloads the initial value and toggles the trigger signal provided to the TGs.

The trigger signal, shown in Figure 21-51 as a square wave, illustrates the different trigger event types for the TGs (for example, rising edge, falling edge, and both edges).

Figure 21-51. Tick Counter Operation



This register is shown in Figure 21-52 and described in Table 21-35.

Figure 21-52. Tick Count Register (TICKCNT) [offset = 90h]

31	30	29	28	27	16
TICKENA	RELOAD	CLKCTRL		Reserved	
R/W-0	R/S-0	R/W-0		R-0	

15	0
TICKVALUE	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; S = Set; -*n* = value after reset

Table 21-35. Tick Count Register (TICKCNT) Field Descriptions

Bit	Field	Value	Description
31	TICKENA		Tick counter enable.
		0	The internal tick counter is disabled. The counter value remains unchanged.
			Note: When the tick counter is disabled, the trigger signal is forced low.
		1	The internal tick counter is enabled and is clocked by the clock source selected by CLKCTRL. When TICKENA goes from 0 to 1, the tick counter is automatically loaded with the contents of TICKVALUE.
30	RELOAD		Pre-load the tick counter. RELOAD is a set-only bit; writing a 1 to it reloads the tick counter with the value stored in TICKVALUE. Reading RELOAD always returns a 0.
			Note: When the tick counter is reloaded by the RELOAD bit, the trigger signal is not toggled.
29-28	CLKCTRL		Tick counter clock source control. CLKCTRL defines the clock source that is used to clock the internal tick counter.
		0	SPICLK of data word format 0 is selected as the clock source of the tick counter.
		1h	SPICLK of data word format 1 is selected as the clock source of the tick counter.
		2h	SPICLK of data word format 2 is selected as the clock source of the tick counter.
		3h	SPICLK of data word format 3 is selected as the clock source of the tick counter.
27-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	TICKVALUE	0-FFFFh	Initial value for the tick counter. TICKVALUE stores the initial value for the tick counter. The tick counter is loaded with the contents of TICKVALUE every time an underflow condition occurs and every time the RELOAD flag is set by the host.

21.8.32 Last TG End Pointer (LTGPEND)

Figure 21-53. Last TG End Pointer (LTGPEND) [offset = 94h]

31 29	28 24	23 16
Reserved	TG_IN_SERVICE	Reserved
R-0	R-0	R-0

15	14 8	7 0
Rsvd	LPEND	Reserved
R-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21-36. Last TG End Pointer (LTGPEND) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return 0. Writes have no effect.
28-24	TG_IN_SERVICE		The TG number currently being serviced by the sequencer. These bits indicate the current TG that is being serviced. This field can generally be used for code debugging.
		0	No TG is being serviced by the sequencer.
		1h	TG0 is being serviced by the sequencer.
		:	:
		10h	TG15 is being serviced by the sequencer.
			Note: The number of transfer groups varies by device.
		11h-1Fh	Invalid values.
23-15	Reserved	0	Reads return 0. Writes have no effect.
14-8	LPEND	0-7Fh	Last TG end pointer. Usually the TG end address (PEND) is inherently defined by the start value of the starting pointer of the subsequent TG (PSTART). The TG ends one word before the next TG starts (PEND[x] = PSTART[x+1] - 1). For a full configuration of MibSPI, the last TG has no subsequent TG, that is, no end address is defined. Therefore, LPEND has to be programmed to specify explicitly the end address of the last TG. **Note: When using all 8 transfer groups, program the LPEND bits to define the end of the last transfer group. When using less than 8 transfer groups, leave the LPEND bits programmed to point to the end of the buffer and create a dummy transfer group that defines the end of your last intentional transfer group and occupies all the remaining buffer space.**
7-0	Reserved	0	Reads return 0. Writes have no effect.

21.8.33 TGx Control Registers (TGxCTRL)

Each TG can be configured by one dedicated control register. The following register description shows one control register(x) that is identical for all TGs. For example, the control register for TG 2 is named TG2CTRL and is located at *base address + 98h + 4 × 2*. The actual number of available control registers varies by device.

Figure 21-54. MibSPI TG Control Registers (TGxCTRL) [offsets = 98h-D4h]

31	30	29	28	27	24
TGENA	ONESHOT	PRST	TGTD	Reserved	
R/W-0	R/W-0	R/W-0	R-0	R-0	

23	20	19	16
TRIGEVT		TRIGSRC	
R/W-0		R/W-0	

15	14	8	7	6	0
Rsvd	PSTART		Rsvd	PCURRENT	
R-0	R/W-0		R-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 21-37. TG Control Registers (TGxCTRL) Field Descriptions

Bit	Field	Value	Description
31	TGENA		TGx enable. If the correct event (TRIGEVTx) occurs at the selected source (TRIGSRCx) a group transfer is initiated if no higher priority TG is in active transfer mode or if one or more higher-priority TGs are in transfer-suspend mode. Disabling a TG while a transfer is ongoing will finish the ongoing word transfer but not the whole group transfer.
		0	TGx is disabled.
		1	TGx is enabled.
30	ONESHOTx		Single transfer for TGx.
		0	TGx initiates a transfer every time a trigger event occurs and TGENA is set.
		1	A transfer from TGx will be performed only once (one shot) after a valid trigger event at the selected trigger source. After the transfer is finished the TGENAx control bit will be cleared and therefore no additional transfer can be triggered before the host enables the TG again. This one shot mode ensures that after one group transfer the host has enough time to read the received data and to provide new transmit data.
29	PRSTx		TGx pointer reset mode. Configures the way to resolve trigger events during an ongoing transfer. This bit is meaningful only for level-triggered TGs. Edge-triggered TGs cannot be restarted before their completion by another edge. The PRST bit will have no effect on this behavior. **Note: When the PRST bit is set, if the buffer being transferred at the time of a new trigger event is a LOCK or CSHOLD buffer, then only after finishing those transfers, the TG will be restarted. This means that even if the TG is re-triggered, the TG will only be restarted after finishing the transfer of the first non-LOCK or non-CSHOLD buffer.** This means that TX control fields such as LOCK and CSHOLD have higher priority over anything else. They have the capability to delay the restart of the TG even if it is re-triggered when PRST is 1.
		0	If a trigger event occurs during a transfer from TGx, the event is ignored and is not stored internally. The TGx transfer has priority over additional trigger events.
		1	The TGx pointer (PCURRENTx) will be reset to the start address (PSTARTx) when a valid trigger event occurs at the selected trigger source while a transfer from the same TG is ongoing. Every trigger event resets PCURRENTx no matter whether the concerned TG is in transfer mode or not. The trigger events have priority over the ongoing transfer.
28	TGTDx		TG triggered.
		0	TGx has not been triggered or is no longer waiting for service.
		1	TGx has been triggered and is either currently being serviced or waiting for servicing.
27-24	Reserved	0	Reads return 0. Writes have no effect.

Table 21-37. TG Control Registers (TGxCTRL) Field Descriptions (continued)

Bit	Field	Value	Description	
23-20	TRIGEVTx		Type of trigger event. A level-triggered TG can be stopped by de-activating the level trigger. However, the following restrictions apply. • Once the transfer of a buffer with CSHOLD or LOCK bit set starts, deactivating the trigger level does not stop the transfer until the sequencer completes the transfer of the next non-CSHOLD or non-LOCK buffer in the same TG. • Once the last buffer in a TG is pre-fetched, de-activating the trigger level does not stop the transfer group until the last buffer transfer is completed. This means even if the trigger level is deactivated at the beginning of the penultimate (one-before-last) buffer transfer, the sequencer continues with the same TG until it is completed.	
		0	never	Never trigger TGx. This is the default value after reset.
		1h	rising edge	A rising edge (0 to 1) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx.
		2h	falling edge	A falling edge (1 to 0) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx.
		3h	both edges	Rising and falling edges at the selected trigger source (TRIGSRCx) initiates a transfer for TGx.
		4h	Rsvd	Reserved
		5h	high-active	While the selected trigger source (TRIGSRCx) is at a logic high level (1) the group transfer is continued and at the end of one group transfer restarted at the beginning. If the logic level changes to low (0) during an ongoing group transfer, the whole group transfer will be stopped. **Note: If ONESHOTx is set the transfer is performed only once.**
		6h	low-active	While the selected trigger source (TRIGSRCx) is at a logic low level (0) the group transfer is continued and at the end of one restarted at the beginning. If the logic level changes to high (1) during an ongoing group transfer, the whole group transfer will be stopped. **Note: If ONESHOTx is set the transfer is performed only once.**
		7h	always	A repetitive group transfer will be performed. **Note: By setting the TRIGSRC to 0, the TRIGEVT to 7h (ALWAYS), and the ONESHOTx bit to 1, software can trigger this TG. Upon setting the TGENA bit, the TG is immediately triggered.** **Note: If ONESHOTx is set the transfer is performed only once.**
		8h-Fh	Rsvd	Reserved

Table 21-37. TG Control Registers (TGxCTRL) Field Descriptions (continued)

Bit	Field	Value	Description	
19-16	TRIGSRCx		Trigger source. After reset, the trigger sources of all TGs are disabled.	
		0	Disabled	
		1h	EXT0	External trigger source 0. The actual source varies per device (for example, HET I/O channel, event pin).
		2h	EXT1	External trigger source 1. The actual source varies per device (for example, HET I/O channel, event pin).
		3h	EXT2	External trigger source 2. The actual source varies per device (for example, HET I/O channel, event pin).
		4h	EXT3	External trigger source 3. The actual source varies per device (for example, HET I/O channel, event pin).
		5h	EXT4	External trigger source 4. The actual source varies per device (for example, HET I/O channel, event pin).
		6h	EXT5	External trigger source 5. The actual source varies per device (for example, HET I/O channel, event pin).
		7h	EXT6	External trigger source 6. The actual source varies per device (for example, HET I/O channel, event pin).
		8h	EXT7	External trigger source 7. The actual source varies per device (for example, HET I/O channel, event pin).
		9h	EXT8	External trigger source 8. The actual source varies per device (for example, HET I/O channel, event pin).
		Ah	EXT9	External trigger source 9. The actual source varies per device (for example, HET I/O channel, event pin).
		Bh	EXT10	External trigger source 10. The actual source varies per device (for example, HET I/O channel, event pin).
		Ch	EXT11	External trigger source 11. The actual source varies per device (for example, HET I/O channel, event pin).
		Dh	EXT12	External trigger source 12. The actual source varies per device (for example, HET I/O channel, event pin).
		Eh	EXT13	External trigger source 13. The actual source varies per device (for example, HET I/O channel, event pin).
		Fh	TICK	Internal periodic event trigger. The tick counter can initiate periodic group transfers.
15	Reserved	0	Reads return 0. Writes have no effect.	
14-8	PSTARTx	0-7Fh	TG start address. PSTARTx stores the start address of the corresponding TG. The corresponding end address is inherently defined by the subsequent TG's start address minus one (PENDx[TGx] = PSTARTx[TGx+1] - 1). PSTARTx is copied into PCURRENTx when: • The TG is enabled • The end of the TG is reached during a transfer • A trigger event occurs while PRST is set to 1	
7	Reserved	0	Reads return 0. Writes have no effect.	
6-0	PCURRENTx	0-7Fh	Pointer to current buffer. PCURRENT is read-only. PCURRENTx stores the address (0...127) of the buffer that corresponds to this TG. If the TG switches from active transfer mode to suspend to wait, PCURRENTx contains the address of the currently suspended word. After the TG resumes from suspend to wait mode, the next buffer will be transferred; that is, no buffer data is transferred because of suspend to wait mode.	

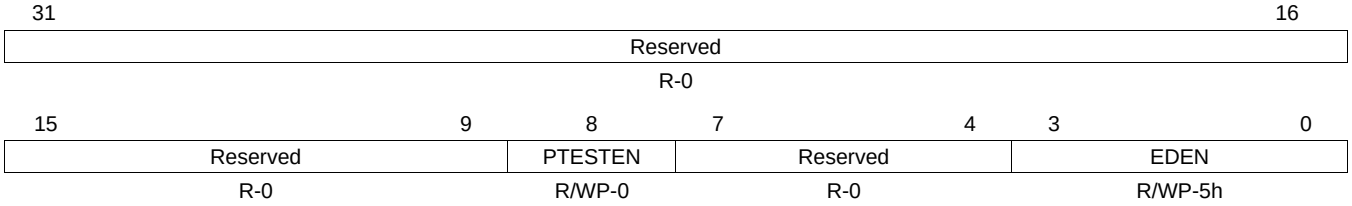
NOTE: Register bits vary by device

TG0 has the highest priority and TG15 has the lowest priority. Under the following conditions a lower priority TG cannot be interrupted by a higher priority TG.

1. When there is a CSHOLD or LOCK buffer, until the completion of the next buffer transfer which is a non-CSHOLD or non-LOCK buffer.
2. Once the last word in a TG is pre-fetched.

21.8.34 Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL)

Figure 21-55. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) [offset = 120h]



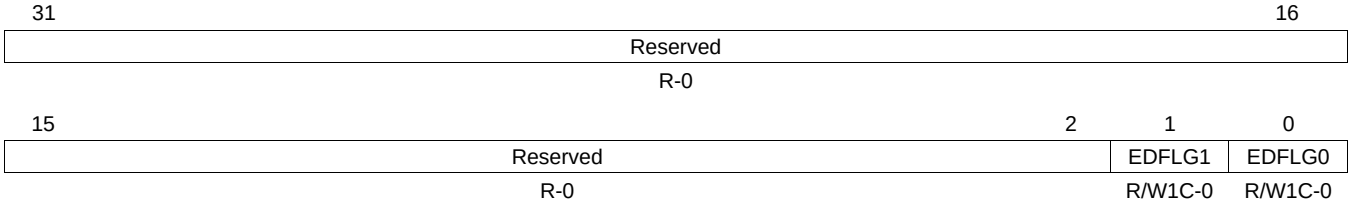
LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 21-38. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	PTESTEN		Parity memory test enable. This bit maps the parity bits corresponding to multi-buffer RAM locations into the peripheral RAM frame to make them accessible by the CPU. See Section 21.10 for further details about parity memory testing.
		0	Parity bits are not memory-mapped.
		1	Parity bits are memory-mapped.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EDEN		Error detection enable. These bits enable parity error detection.
		5h	Parity error detection logic (default) is disabled.
		All other values	Parity error detection logic is enabled.
			Note: It is recommended to write a 1010 to enable error detection, to guard against a soft error from disabling parity error detect.

21.8.35 Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT)

Figure 21-56. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) [offset = 124h]



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 21-39. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	EDFLG1		Uncorrectable parity error detection flag. This flag indicates if a parity error occurred in the RXRAM. **Note: Reading the UERRADDR1 register clears the EDFLG1 bit.**
		0	Read: No error has occurred. Write: Writing a 0 to this bit has no effect.
		1	Read: An error was detected and the address is captured in the UERRADDR1 register. Write: The bit is cleared to 0.
0	EDFLG0		Uncorrectable parity error detection flag. This flag indicates if a parity error occurred in the TXRAM. **Note: Reading the UERRADDR0 register clears the EDFLG0 bit.**
		0	Read: No error has occurred. Write: Writing a 0 to this bit has no effect.
		1	Read: An error was detected and the address is captured in the UERRADDR0 register. Write: The bit is cleared to 0.

21.8.36 RXRAM Uncorrectable Parity Error Address Register (UERRADDR1)

Figure 21-57. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) [offset = 128h]

31	16
Reserved	
R-0	

15	10	9	0
Reserved		OVERADDR1	
R-0		R/C-x	

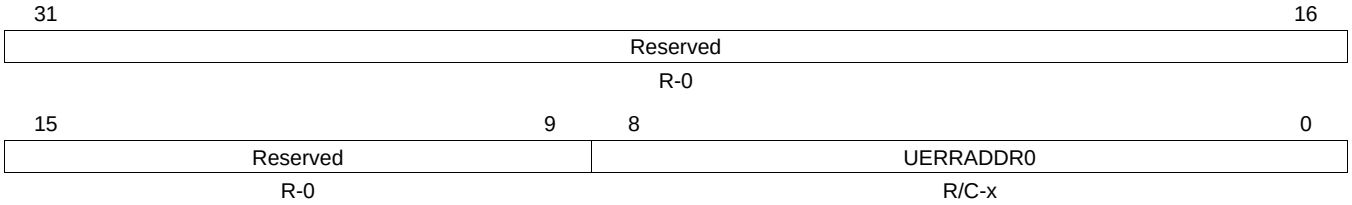
LEGEND: R = Read only; C = Clear; -*n* = value after reset

Table 21-40. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	OVERADDR1	200h-3FFh	Uncorrectable parity error address for RXRAM. This register holds the address where a parity error is generated while reading RXRAM. Only the CPU can read from RXRAM locations. The address captured is byte-aligned. This error address is frozen from being updated until it is read by the CPU. The offset address of RXRAM varies from 200h-3FFh. The register does not clear its contents during or after module-level reset, system-level reset or even power-on reset. A read operation to this register clears its contents to the default value 200h. After a power-on reset the contents will be unpredictable. A read operation can be performed after power-up to keep the register at its default value, if required. However, the contents of this register are meaningful only when EDFLG1 is set to 1. **Note: A read of the UERRADDR1 register will clear EDFLG1 in the UERRSTAT register. However, in emulation mode when the SUSPEND signal is high, a read from the UERRADDR1 register does not clear EDFLG1.**

21.8.37 TXRAM Uncorrectable Parity Error Address Register (UERRADDR0)

Figure 21-58. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) [offset = 12Ch]



LEGEND: R = Read only; C = Clear; -*n* = value after reset

Table 21-41. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	UERRADDR1	0-1FFh	Uncorrectable parity error address for TXRAM. This register holds the address where a parity error is generated while reading from TXRAM. The TXRAM can be read either by CPU or by the MibSPI sequencer logic for transmission. The address captured is byte- aligned. This error address is frozen from being updated until it is read by the CPU. The offset address of TXRAM varies from 0-1FFh. The register does not clear its contents during or after module-level reset, system-level reset, or even power-on reset. A read operation to this register clears its contents to all 0s. After a power-on reset, the contents of this register will be unpredictable. A read operation can be performed after power-up to clear the this register's contents, if required. However, the contents of this register are meaningful only when EDFLG0 is set to 1. **Note: A read from the UERRADDR0 register will clear EDFLG0 in the UERRSTAT register. However, in emulation mode when the SUSPEND signal is high, a read from the UERRADDR0 register does not clear EDFLG0.**

21.8.38 RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR)

In multi-buffer mode, if a particular RXRAM location is written by the MibSPI sequencer logic after the completion of a new transfer when that location already contains valid data, the RX_OVR bit will be set to 1 while the data is being written. The RXOVRN_BUF_ADDR register captures the address of the RXRAM location for which a receiver overrun condition occurred.

Figure 21-59. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) [offset = 130h]

31	16
Reserved	
R-0	

15	10	9	0
Reserved		RXOVRN_BUF_ADDR	
R-0		R-200h	

LEGEND: R = Read only; -*n* = value after reset

Table 21-42. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	RXOVRN_BUF_ADDR	200h-3FCh	Address in RXRAM at which an overwrite occurred. This address value will show only the offset address of the RAM location in the multi-buffer RAM address space. Refer to the device-specific data sheet for the actual absolute address of RXRAM. This word-aligned address can vary from 200h-3FCh. Contents of this register are valid only when any of the INTVECT0 or INTVECT1 and SPIFLG registers show an RXOVRN error vector while in multi-buffer mode. If there are multiple overrun errors, then this register holds the address of first overrun address until it is read. **Note: Reading this register clears the RXOVRN interrupt flag in the SPIFLG register and the TGINTVECTx.** **Note: Receiver overrun errors in multi-buffer mode can be completely avoided by using the SUSPEND until RXEMPTY feature, which can be programmed into each buffer of any TG. However, using the SUSPEND until RXEMPTY feature will make the sequencer wait until the current RXRAM location is read by the VBUS master before it can start the transfer for the same buffer location again. This may affect the overall throughput of the SPI transfer. By enabling the interrupt on RXOVRN in multi-buffer mode, the user can rely on interrupts to know if a receiver overrun has occurred. The address of the overrun in RXRAM is indicated in this RXOVRN_BUF_ADDR register.**

21.8.39 I/O-Loopback Test Control Register (IOLPBKTSTCR)

This register controls test mode for I/O pins. It also controls whether loop-back should be digital or analog. In addition, it contains control bits to induce error conditions into the module. These are to be used only for module testing.

All of the control/status bits in this register are valid only when the IOLPBKTSTENA field is set to Ah.

Figure 21-60. I/O-Loopback Test Control Register (IOLPBKTSTCR) [offset = 134h]

31						25	24
Reserved							SCS_FAIL_FLG
R-0							R/W1C-0
23		21	20	19	18	17	16
Reserved			CTRL_BITERR	CTRL_DESYNC	CTRL_PARERR	CTRL_TIMEOUT	CTRL_DLENERR
R-0			R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0
15			12	11			8
Reserved				IOLPBKTSTENA			
R-0				R/WP-0			
7	6	5		3	2	1	0
Reserved		ERR_SCS_PIN			CTRL_SCS_PIN_ERR	LPBKTYPE	RXPENA
R-0		R/WP-0			R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; WP = Write in privilege mode only; -*n* = value after reset

Table 21-43. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	SCS_FAIL_FLG		Bit indicating a failure on $\overline{\text{SPICS}}$ pin compare during analog loopback.
		0	Read: No miscompares occurred on any of the eight chip select pins (vs. the internal chip select number CSNR during transfers). Write: Writing a 0 to this bit has no effect.
		1	Read: A comparison between the internal CSNR field and the analog looped-back value of one or more of the $\overline{\text{SPICS}}$ pins failed. A stuck-at fault is detected on one of the $\overline{\text{SPICS}}$. Comparison is done only on the pins that are configured as functional and during transfer operation. Write: This flag bit is cleared.
23-21	Reserved	0	Reads return 0. Writes have no effect.
20	CTRL_BITERR		Controls inducing of BITERR during I/O loopback test mode.
		0	Do not interfere with looped-back data.
		1	Induces bit errors by inverting the value of the incoming data during loopback.
19	CTRL_DESYNC		Controls inducing of the desync error during I/O loopback test mode.
		0	Do not cause a desync error.
		1	Induce a desync error by forcing the incoming $\overline{\text{SPIENA}}$ pin (if functional) to remain 0 even after the transfer is complete. This forcing will be retained until the kernel reaches the idle state.
18	CTRL_PARERR		Controls inducing of the parity errors during I/O loopback test mode.
		0	Do not cause a parity error.
		1	Induce a parity error by inverting the polarity of the parity bit.
17	CTRL_TIMEOUT		Controls inducing of the timeout error during I/O loopback test mode.
		0	Do not cause a timeout error.
		1	Induce a timeout error by forcing the incoming $\overline{\text{SPIENA}}$ pin (if functional) to remain 1 when transmission is initiated. The forcing will be retained until the kernel reaches the idle state.

Table 21-43. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions (continued)

Bit	Field	Value	Description
16	CTRL_DLENERR		Controls inducing of the data length error during I/O loopback test mode.
		0	Do not cause a data-length error.
		1	Induce a data-length error.
			Master mode: The $\overline{\text{SPIENA}}$ pin (if functional) is forced to 1 when the module starts shifting data.
			Slave mode: The incoming $\overline{\text{SPICS}}$ pin (if functional) is forced to 1 when the module starts shifting data.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	IOLPBKSTENA		Module I/O loopback test enable key.
		Ah	Enable I/O loopback test mode.
		All other values	Disable I/O loopback test mode.
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-3	ERR_SCS_PIN		Inject error on chip-select pin number x. The value in this field is decoded as the number of the chip select pin on which to inject an error. During analog loopback, if CTRL SCS PIN ERR bit is set to 1, then the chip select pin selected by this field is forced to the opposite of its value in the CSNR.
		0	Select $\overline{\text{SPICS}}$[0] for injecting error.
		1h	Select $\overline{\text{SPICS}}$[1] for injecting error.
		2h	Select $\overline{\text{SPICS}}$[2] for injecting error.
		3h	Select $\overline{\text{SPICS}}$[3] for injecting error.
		4h-7h	Reserved
2	CTRL_SCS_PIN_ERR		Enable/disable the injection of an error on the $\overline{\text{SPICS}}$ pins. The individual $\overline{\text{SPICS}}$ pins can be chosen using the ERR SCS PIN field.
		0	Disable the $\overline{\text{SPICS}}$ error-inducing logic.
		1	Enable the $\overline{\text{SPICS}}$ error-inducing logic.
1	LPBKTYPE		Module I/O loopback type (analog/digital). See Figure 21-14 for the different types of loopback modes.
		0	Enable Digital loopback when IOLPBKTSTENA = 1010.
		1	Enable Analog loopback when IOLPBKTSTENA = 1010.
0	RXPENA		Enable analog loopback through the receive pin.
			Note: This bit is valid only when LPBK TYPE = 1, which chooses analog loopback mode.
		0	Analog loopback is through the transmit pin.
		1	Analog loopback is through the receive pin.

21.8.40 SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1 for SPIFMT0 and SPIFMT1)

This register provides an extended Prescale values for SPICLK generation to be able to interface with much slower SPI Slaves. This is an extension of SPIFMT0 and SPIFMT1 registers. For example, EPRESCALE_FMT1[7:0] of EXTENDED_PRESCALE1 and PRESCALE1 of SPIFMT1 register will always reflect the same contents. Similarly, EPRESCALE_FMT0[7:0] and PRESCALE0 of SPIFMT0 reflect the same contents.

Figure 21-61. SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1 for SPIFMT0 and SPIFMT1) [offset = 138h]

31	27	26	16
Reserved		EPRESCALE_FMT1	
R-0		R/WP-0	

15	11	10	0
Reserved		EPRESCALE_FMT0	
R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 21-44. SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26-16	EPRESCALE_FMT1	0-7FFh	EPRESCALE_FMT1. Extended Prescale value for SPIFMT1. EPRESCALE_FMT1 determines the bit transfer rate of data format 1 if the SPI/MibSPI is the network master. EPRESCALE_FMT1 is use to derive SPICLK from VCLK. If the SPI is configured as slave, EPRESCALE_FMT1 **does not need** to be configured. These EPRESCALE_FMT1[7:0] bits and PRESCALE1 bits of SPIFMT1 register will point to the same physically implemented register. The clock rate for data format 1 can be calculated as: $BR_{Format1}$ = VCLK / (EPRESCALE_FMT1 + 1) Write: This register field should be written if a SPICLK prescaler of more VCLK/256 is required. This field provides a prescaler of up to VCLK/2048 for SPICLK. Writing to this register field will also get reflected in the PRESCALE1 bits of SPIFMT1 register. Read: Reading this field will reflect the PRESCALE value based on the last written register field, that is, EXTENDED_PRESCALE1[26:16] or SPIFMT1[15:8] register. **Note: If Extended Prescaler is required, it should be ensured that EXTENDED_PRESCALE1 register is programmed after SPIFMT1 register is programmed. This is to ensure that the final SPICLK prescale value is controlled by EXTENDED_PRESCALE1 register when a prescale of more 256 is intended on SPICLK. Writing to PRESCALE1 field of SPIFMT1 will automatically clear EPRESCALE_FMT1[10:8] bits to 000 so that the integrity of PRESCALE value is maintained.**
15-11	Reserved	0	Reads return 0. Writes have no effect.

Table 21-44. SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1) Field Descriptions (continued)

Bit	Field	Value	Description
10-0	EPRESCALE_FMT0	0-7FFh	EPRESCALE_FMT0. Extended Prescale value for SPIFMT0. EPRESCALE_FMT0 determines the bit transfer rate of data format 0 if the SPI/MibSPI is the network master. EPRESCALE_FMT0 is use to derive SPICLK from VCLK. If the SPI is configured as slave, EPRESCALE_FMT0 **does not need** to be configured. These EPRESCALE_FMT0[7:0] bits and PRESCALE0 bits of SPIFMT0 register will point to the same physically implemented register. The clock rate for data format 0 can be calculated as: $BR_{Format0}$ = VCLK / (EPRESCALE_FMT0 + 1) Write: This register field should be written if a SPICLK prescaler of more VCLK/256 is required. This field provides a prescaler of up to VCLK/2048 for SPICLK. Writing to this register field will also get reflected in the PRESCALE0 bits of SPIFMT0 register. Read: Reading this field will reflect the PRESCALE value based on the last written register field, that is, EXTENDED_PRESCALE0[10:0] or SPIFMT0[15:8] register. **Note: If Extended Prescaler is required, it should be ensured that EXTENDED_PRESCALE1 register is programmed after SPIFMT0 register is programmed. This is to ensure that the final SPICLK prescale value is controlled by EXTENDED_PRESCALE1 register when a prescale of more 256 is intended on SPICLK. Writing to PRESCALE0 field of SPIFMT0 will automatically clear EPRESCALE_FMT0[10:8] bits to 000 so that the integrity of PRESCALE value is maintained.**

21.8.41 SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2 for SPIFMT2 and SPIFMT3)

This register provides an extended Prescale values for SPICLK generation to be able to interface with much slower SPI Slaves. This is an extension of SPIFMT2 and SPIFMT3 registers. For example, EPRESCALE_FMT3[7:0] of EXTENDED_PRESCALE2 and PRESCALE3 of SPIFMT3 register will always reflect the same contents. Similarly, EPRESCALE_FMT2[7:0] and PRESCALE2 of SPIFMT2 reflect the same contents.

Figure 21-62. SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2 for SPIFMT2 and SPIFMT3) [offset = 13Ch]

31	27	26	16
Reserved		EPRESCALE_FMT3	
R-0		R/WP-0	

15	11	10	0
Reserved		EPRESCALE_FMT2	
R-0		R/WP-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset

Table 21-45. SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26-16	EPRESCALE_FMT3	0-7FFh	EPRESCALE_FMT3. Extended Prescale value for SPIFMT3. EPRESCALE_FMT3 determines the bit transfer rate of data format 3 if the SPI/MibSPI is the network master. EPRESCALE_FMT3 is use to derive SPICLK from VCLK. If the SPI is configured as slave, EPRESCALE_FMT3 **does not need** to be configured. These EPRESCALE_FMT3[7:0] bits and PRESCALE3 bits of SPIFMT3 register will point to the same physically implemented register. The clock rate for data format 1 can be calculated as: $BR_{Format3}$ = VCLK / (EPRESCALE_FMT3 + 1) Write: This register field should be written if a SPICLK prescaler of more VCLK/256 is required. This field provides a prescaler of up to VCLK/2048 for SPICLK. Writing to this register field will also get reflected in the PRESCALE3 bits of SPIFMT3 register. Read: Reading this field will reflect the PRESCALE value based on the last written register field, that is, EXTENDED_PRESCALE3[26:16] or SPIFMT3[15:8] register. **Note: If Extended Prescaler is required, it should be ensured that EXTENDED_PRESCALE2 register is programmed after SPIFMT3 register is programmed. This is to ensure that the final SPICLK prescale value is controlled by EXTENDED_PRESCALE2 register when a prescale of more 256 is intended on SPICLK. Writing to PRESCALE3 field of SPIFMT3 will automatically clear EPRESCALE_FMT3[10:8] bits to 000 so that the integrity of PRESCALE value is maintained.**
15-11	Reserved	0	Reads return 0. Writes have no effect.

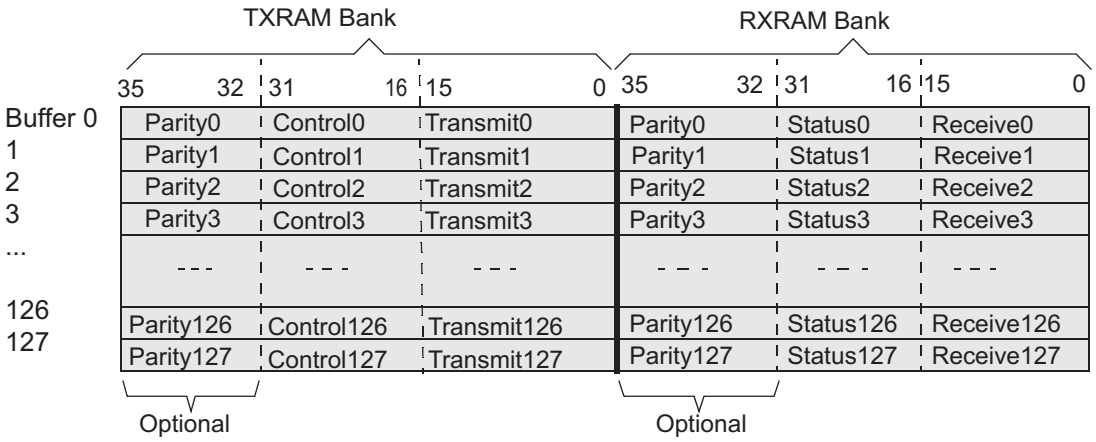
Table 21-45. SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2) Field Descriptions (continued)

Bit	Field	Value	Description
10-0	EPRESCALE_FMT2	0-7FFh	EPRESCALE_FMT2. Extended Prescale value for SPIFMT2. EPRESCALE_FMT2 determines the bit transfer rate of data format 2 if the SPI/MibSPI is the network master. EPRESCALE_FMT2 is use to derive SPICLK from VCLK. If the SPI is configured as slave, EPRESCALE_FMT2 **does not need** to be configured. These EPRESCALE_FMT2[7:0] bits and PRESCALE2 bits of SPIFMT2 register will point to the same physically implemented register. The clock rate for data format 0 can be calculated as: $BR_{Format2}$ = VCLK / (EPRESCALE_FMT2 + 1) Write: This register field should be written if a SPICLK prescaler of more VCLK/256 is required. This field provides a prescaler of up to VCLK/2048 for SPICLK. Writing to this register field will also get reflected in the PRESCALE2 bits of SPIFMT2 register. Read: Reading this field will reflect the PRESCALE value based on the last written register field, that is, EXTENDED_PRESCALE2[10:0] or SPIFMT2[15:8] register. **Note: If Extended Prescaler is required, it should be ensured that EXTENDED_PRESCALE2 register is programmed after SPIFMT2 register is programmed. This is to ensure that the final SPICLK prescale value is controlled by EXTENDED_PRESCALE2 register when a prescale of more 256 is intended on SPICLK. Writing to PRESCALE2 field of SPIFMT2 will automatically clear EPRESCALE_FMT2[10:8] bits to 000 so that the integrity of PRESCALE value is maintained.**

21.9 Multi-Buffer RAM

The multi-buffer RAM is used for holding transit & received data, control and status information. The multi-buffer RAM contains two banks of up to 128 32-bit words for a maximum configuration. One bank (TXRAM) contains entries for transmit data (replicating the SPIDAT1 register). The other bank (RXRAM) contains received data (replicating the SPIBUF register). The buffers can be partitioned into multiple TGs, each containing a programmable number of buffers. Each of the buffers can be subdivided into 16-bit transmit field, 16-bit receive field, 16-bit control field, and 16-bit status field, as displayed in Figure 21-63. A 4-bit parity field per word is also included in each bank of RAM.

Figure 21-63. Multi-Buffer RAM Configuration



All fields can be read and written with 8-bit, 16-bit, or 32-bit accesses.

The transmit fields can be written and read in the address range 000h to 1FFh. The transmit words contain data and control fields.

The receive RAM fields are read-only and can be accessed through the address range 200h to 3FCh. The receive words contain data and status fields.

The chip select number bit field CSNR[7:0] of the control field for a given word is mirrored into the corresponding receive-buffer status field after transmission.

The Parity is automatically calculated and copied to Parity location

NOTE: Please refer to the specific device datasheet for the actual number of transmit and receive buffers.

Write to unimplemented buffer is overwriting the corresponding implemented buffer. In MIBSPI, if the RAM SIZE specified is 32 buffers, write to 33rd buffer overwrites 1st buffer, write to 34th buffer overwrites 2st buffer and so on.

21.9.1 Multi-Buffer RAM Auto Initialization

When the MIBSPI is out of reset mode, auto initialization of multi-buffer RAM starts. The application code must check for BUFINITACTIVE bit to be 0 (Multi-buffer RAM initialization is complete) before configuring multi-buffer RAM.

Besides the default auto initialization after reset, the auto-initialization sequence can also be done by:

1. Enable the global hardware memory initialization key by programming a value of 1010b to the bits [3:0] of the MINITGCR register of the System module.
2. Set the control bit for the multi-buffer RAM in the MSINENA System module register. This bit is device-specific for each memory that support auto-initialization. Please refer to the device datasheet to identify the control bit for the multi-buffer RAM. This starts the initialization process. The BUFINITACTIVE bit will get set to reflect that the initialization is ongoing.
3. When the memory initialization is completed, the corresponding status bit in the MINISTAT register will be set. Also, the BUFINITACTIVE bit will get cleared.
4. Disable the global hardware memory initialization key by programming a value of 0101 to the bits [3:0] of the MINITGCR register of the System module.

Please refer to the Architecture User Guide for more details on the memory auto-initialization process.

NOTE: During Auto Initialization process, all the Multi-buffer mode registers (except MIBSPIE) will be reset to their default values. So, it should be ensured that Auto Initialization is completed before configuring the Multi-buffer mode registers.

21.9.2 Multi-buffer RAM Register Summary

This section describes the Multi-buffer RAM control and transmit-data fields of each word of TXRAM, and the status and receive-data fields of each word of RXRAM. The base address for multi-buffer RAM is FF0E 0000h for MibSPI1 RAM.

Table 21-46. Multi-buffer RAM Register Summary

Offset	Acronym	Register Description	Section
Base + 000h-1FFh	TXRAM	Multi-buffer RAM Transmit Data Register	Section 21.9.3
Base + 200h-3FFh	RXRAM	Multi-buffer RAM Receive Buffer Register	Section 21.9.4

21.9.3 Multi-buffer RAM Transmit Data Register (TXRAM)

Each word of TXRAM is a transmit-buffer register.

NOTE: Writing to only the control fields, bits 28 through 16, does not initiate any SPI transfer in master mode. This feature can be used to set up SPICLK phase or polarity before actually starting the transfer by only updating the DFSEL bit field to select the required phase and polarity combination.

Figure 21-64. Multi-buffer RAM Transmit Data Register (TXRAM) [offset = Base + 000h-1FFh]

31–29	28	27	26	25–24	23–16
BUFMODE	CSHOLD	LOCK	WDEL	DFSEL	CSNR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15–0
TXDATA
R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21-47. Multi-buffer RAM Transmit Data Register (TXRAM) Field Descriptions

Bit	Field	Value	Description
31-29	BUFMODE		Specify conditions that are recognized by the sequencer to initiate transfers of each buffer word. When one of the "skip" modes is selected, the sequencer checks the buffer status every time it reads from this buffer. If the current buffer status (TXFULL, RXEMPTY) does not match, the buffer is skipped without a data transfer. When one of the "suspend" modes is selected, the sequencer checks the buffer status when it reads from this buffer. If TXFULL and/or RXEMPTY do not match, the sequencer waits until a match occurs. No data transfer is initiated until the status condition of this buffer changes.
		0	**disabled.** The buffer is disabled
		1h	**skip single-transfer mode.** Skip this buffer until the corresponding TXFULL flag is set (new transmit data is available).
		2h	**skip overwrite-protect mode.** Skip this buffer until the corresponding RXEMPTY flag is set (new receive data can be stored in RXDATA without data loss).
		3h	**skip single-transfer overwrite-protect mode**. Skip this buffer until both of the corresponding TXFULL and RXEMPTY flags are set. (new transmit data available and previous data received by the host).
		4h	**continuous mode.** Initiate a transfer each time the sequencer checks this buffer. Data words are retransmitted if the buffer has not been updated. Receive data is overwritten, even if it has not been read.
		5h	**suspend single-transfer mode.** Suspend-to-wait until the corresponding TXFULL flag is set (the sequencer stops at the current buffer until new transmit data is written in the TXDATA field).
		6h	**suspend overwrite-protect mode.** Suspend-to-wait until the corresponding RXEMPTY flag is set (the sequencer stops at the current buffer until the previously-received data is read by the host.
		7h	**suspend single-transfer overwrite-protect mode.** Suspend-to-wait until the corresponding TXFULL and RXEMPTY flags are set (the sequencer stops at the current buffer until new transmit data is written into the TXDATA field and the previously-received data is read by the host).
28	CSHOLD		Chip select hold mode. The CSHOLD bit is supported in master mode only, it is ignored in slave mode. CSHOLD defines the behavior of the chip select line at the end of a data transfer.
		0	The chip select signal is deactivated at the end of a transfer after the T2CDELAY time has passed. If two consecutive transfers are dedicated to the same chip select this chip select signal will be deactivated for at least 2VCLK cycles before it is activated again.
		1	The chip select signal is held active at the end of a transfer until a control field with new data and control information is loaded into SPIDAT1. If the new chip select number equals the previous one, the active chip select signal is extended until the end of transfer with CSHOLD cleared, or until the chip-select number changes.

Table 21-47. Multi-buffer RAM Transmit Data Register (TXRAM) Field Descriptions (continued)

Bit	Field	Value	Description
27	LOCK		Lock two consecutive buffer words. Do not allow interruption by TGs with higher priority.
		0	Any higher-priority TG can begin at the end of the current transaction.
		1	A higher-priority TG cannot occur until after the next unlocked buffer word is transferred.
26	WDEL		Enable the delay counter at the end of the current transaction.
			Note: The WDEL bit is supported in master mode only. In slave mode, this bit is ignored.
		0	No delay will be inserted. However, $\overline{\text{SPICS}}$ pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0.
			Note: The duration for which the $\overline{\text{SPICS}}$ pin remains deactivated also depends upon the time taken to supply a new word after completing the shift operation (in compatibility mode). If TXBUF is already full, then the $\overline{\text{SPICS}}$ pin will be deasserted for at least two VCLK cycles (if WDEL = 0).
		1	After a transaction, WDELAY of the corresponding data format will be loaded into the delay counter. No transaction will be performed until the WDELAY counter overflows. The $\overline{\text{SPICS}}$ pins will be de-activated for at least (WDELAY + 2) × VCLK_Period duration.
25-24	DFSEL		Data word format select.
		0	Data word format 0 is selected.
		1h	Data word format 1 is selected.
		2h	Data word format 2 is selected.
		3h	Data word format 3 is selected.
23-16	CSNR	0-FFh	Chip select (CS) number. CSNR defines the chip select pins that will be activated during the data transfer. CSNR is a bit-mask that controls all chip select pins. See Table 21-48.
			Note: If your MibSPI has less than 8 chip select pins, all unused upper bits will be 0. For example, the MiBSPI has 4 chip select pins, if you write FFh to CSNR, the actual number stored in CSNR is 1Fh.
15-0	TXDATA	0-7FFFh	Transfer data. When written, these bits are copied to the shift register if it is empty. If the shift register is not empty, then they are held in TXBUF.
			SPIEN must be set to 1 before this register can be written to. Writing a 0 to SPIEN forces the lower 16 bits of TXDATA to 0.
			A write to this register (or to the TXDATA field only) drives the contents of the CSNR field on the $\overline{\text{SPICS}}$ pins, if the pins are configured as functional pins (automatic chip select, see Section 21.2).
			When this register is read, the contents of TXBUF, which holds the latest data written, will be returned.
			Note: Regardless of the character length, the transmit data should be right-justified before writing to the SPIDAT1 register.

Table 21-48. Chip Select Number Active

CSNR Value	Chip Select Active:			
	CS[3]	CS[2]	CS[1]	CS[0]
0h	No chip select pin is active.			
1h				x
2h			x	
3h			x	x
4h		x		
5h		x		x
6h		x	x	
7h		x	x	x
8h	x			
9h	x			x
Ah	x		x	
Bh	x		x	x
Ch	x	x		
Dh	x	x		x
Eh	x	x	x	
Fh	x	x	x	x
10h				
11h				x
12h			x	
13h			x	x
14h		x		
15h		x		x
16h		x	x	
17h		x	x	x
18h	x			
19h	x			x
1Ah	x		x	
1Bh	x		x	x
1Ch	x	x		
1Dh	x	x		x
1Eh	x	x	x	
1Fh	x	x	x	x

21.9.4 Multi-buffer RAM Receive Buffer Register (RXRAM)

Each word of RXRAM is a receive-buffer register.

Figure 21-65. Multi-buffer RAM Receive Buffer Register (RXRAM) [offset = RAM Base + 200h-3FFh]

31	30	29	28	27	26	25	24
RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARITYERR	TIMEOUT	DLENERR
RS-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	RC-0

23	16
LCSNR	
R-0	

15	0
RXDATA	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; S = Set; -*n* = value after reset

Table 21-49. Multi-buffer Receive Buffer Register (RXRAM) Field Descriptions

Bit	Field	Value	Description
31	RXEMPTY		Receive data buffer empty.When the host reads the RXDATA field or the entire RXRAM register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into RXDATA, and the RXEMPTY flag is cleared.
		0	New data has been received and copied into RXDATA.
		1	No data has been received since the last read of RXDATA.
			This flag gets set to 1 under the following conditions: • Reading the RXDATA portion of the RXRAM register • Writing a 1 to clear the RXINTFLG bit in the SPI Flag Register (SPIFLG) Write-clearing the RXINTFLG bit before reading RXDATA indicates the received data is being ignored. Conversely, RXINTFLG can be cleared by reading the RXDATA portion of RXRAM (or the entire register).
30	RXOVR		Receive data buffer overrun. When a data transfer is completed and the received data is copied into RXBUF while it is already full, RXOVR is set. Overruns always occur to RXBUF, not to RXRAM; the contents of RXRAM are overwritten only after it is read by the Peripheral (VBUSP) master (CPU or other host processor). If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either SPI Flag Register (SPIFLG) or SPIVECTx shows the RXOVRN condition. Two read operations from the RXRAM register are required to reach the overwritten buffer word (one to read RXRAM, which then transfers RXDATA into RXRAM for the second read). **Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.** **Note: A special condition under which RXOVR flag gets set. If both RXRAM and RXBUF are already full and while another buffer receive is underway, if any errors such as TIMEOUT, BITERR and DLEN_ERR occur, then RXOVR in RXBUF and SPI Flag Register (SPIFLG) registers will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receive overrun.**
		0	No receive data overrun condition occurred since last read of the data field.
		1	A receive data overrun condition occurred since last read of the data field.
29	TXFULL		Transmit data buffer full.This flag is a read-only flag. Writing into the SPIDAT0 or SPIDAT1 field while the TX shift register is full will automatically set the TXFULL flag. Once the word is copied to the shift register, the TXFULL flag will be cleared. Writing to SPIDAT0 or SPIDAT1 when both TXBUF and the TX shift register are empty does not set the TXFULL flag.
		0	The transmit buffer is empty; SPIDAT0/SPIDAT1 is ready to accept a new data.
		1	The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data.

Table 21-49. Multi-buffer Receive Buffer Register (RXRAM) Field Descriptions (continued)

Bit	Field	Value	Description
28	BITERR		Bit error. There was a mismatch of internal transmit data and transmitted data.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No bit error occurred.
		1	A bit error occurred. The SPI samples the signal of the transmit pins (master: SIMOx, slave: SOMIx) at the receive point (one-half clock cycle after the transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the BITERR flag is set. Possible reasons for a bit error include noise, an excessively high bit rate, capacitive load, or another master/slave trying to transmit at the same time.
27	DESYNC		Desynchronization of slave device. This bit is valid in master mode only.
			The master monitors the ENA signal coming from the slave device and sets the DESYNC flag if ENA is deactivated before the last reception point or after the last bit is transmitted plus $t_{T2EDELAY}$. If DESYNCENA is set, an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.
			Note: In the Compatibility Mode MibSPI, under some circumstances it is possible for a desync error detected for the previous buffer to be visible in the current buffer. This is because the receive completion flag/interrupt is generated when the buffer transfer is completed. But desynchronization is detected after the buffer transfer is completed. So, if the VBUS master reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition. In multi-buffer mode, the desync flag is always assured to be for the current buffer.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No slave desynchronization is detected.
		1	A slave device is desynchronized.
26	PARITYERR		Parity error. The calculated parity differs from the received parity bit.
			If the parity generator is enabled (selected individually for each buffer) an even or odd parity bit is added at the end of a data word. During reception of the data word, the parity generator calculates the reference parity and compares it to the received parity bit. If a mismatch is detected, the PARITYERR flag is set.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No parity error is detected.
		1	A parity error occurred.
25	TIMEOUT		Time-out because of non-activation of $\overline{\text{SPIENA}}$ pin.
			The SPI generates a time-out when the slave does not respond in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected, the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition, the TIMEOUT flag in the status field of the corresponding buffer and in the SPI Flag Register (SPIFLG) is set.
			Note: This bit is valid only in master mode.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No $\overline{\text{SPIENA}}$ pin time-out occurred.
		1	An $\overline{\text{SPIENA}}$ signal time-out occurred.
24	DLENERR		Data length error flag.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No data-length error has occurred.
		1	A data length error has occurred.
23-16	LCSNR	0-FFh	Last chip select number. LCSNR in the status field is a copy of CSNR in the corresponding control field. It contains the chip select number that was activated during the last word transfer.
15-0	RXDATA	0-FFFFh	SPI receive data. This is the received word, transferred from the receive shift-register at the end of a transfer. Regardless of the programmed character length and the direction of shifting, the received data is stored right-justified in the register.

21.10 Parity Memory

The parity portion of multi-buffer RAM is not accessible by the CPU during normal operating modes. However, each read or write operation to the control/data/status portion of the multi-buffer RAM causes reads/writes to the parity portion as well.

- Each write to the multi-buffer RAM (either from the Peripheral interface or by the MibSPI itself) causes a write operation to the parity portion of RAM simultaneously to update the equivalent parity bits.
- Each read operation from the multi-buffer RAM (either from the Peripheral interface or by the MibSPI itself) causes a read operation from the parity portion of the RAM for parity comparison purpose.
- Reads/Writes to multi-buffer RAM can either be caused by any CPU accesses or by the sequencer logic of MibSPI itself.
- Incase of Parity error ESM module is notified to generate MIBSPI Parity ESM interrupt. User can check the error status and address location captured in the UERRSTAT and UERRADDRx registers respectively.

For testing the parity portion of the multi-buffer RAM, which is a 4-bit field per word address (1 bit per byte), a separate parity memory test mode is available. Parity memory test mode can be enabled and disabled by the PTESTEN bit in the UERRCTRL register.

During the parity test mode, the parity locations are addressable at the address between RAM_BASE_ADDR + 0x400h and RAM_BASE_ADDR + 0x7FFh. Each location corresponds, sequentially, to each TXRAM word, then to each RXRAM word. See Figure 21-66 for a diagram of the memory map of parity memory during normal operating mode and during parity test mode.

During parity test mode, after writing the data/control portion of the RAM, the parity locations can be written with incorrect parity bits to intentionally cause parity errors.

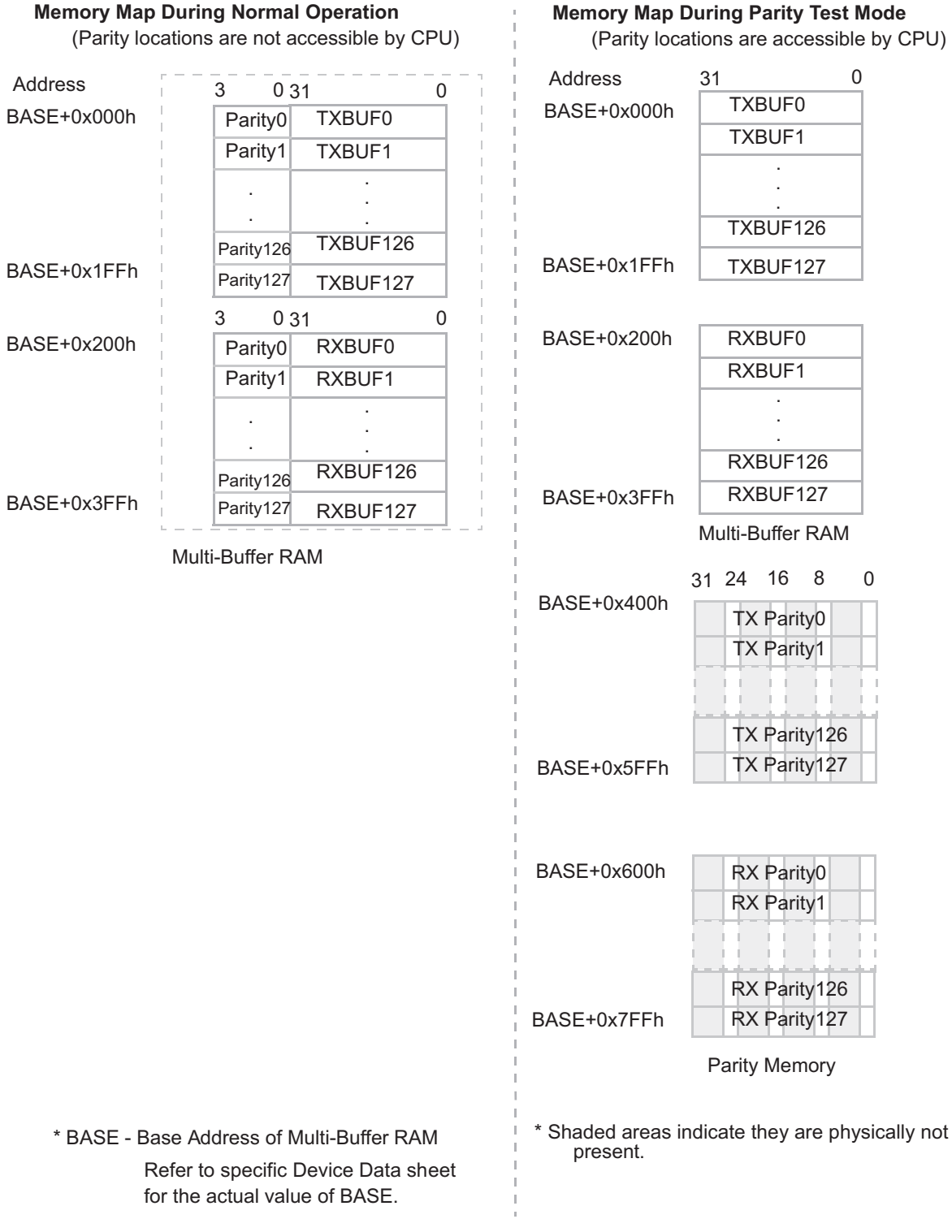
See the device-specific data sheet to get the actual base address of the multi-buffer RAM.

NOTE: The RX_RAM_ACCESS bit can also be set to 1 during the parity test mode to be enable writes to RXRAM locations. Both parity RAM testing and RXRAM testing can be done together.

There are 4 bits of parity corresponding to each of the 32-bit multi-buffer locations. Individual bits in the parity memory are byte-addressable in parity test mode. See the example in Figure 21-67 for further details.

NOTE: Polarity of the parity (odd/even) varies by device. In some devices, a control register in the system module can be used to select odd or even parity.

Figure 21-66. Memory Map for Parity Locations During Normal and Test Mode

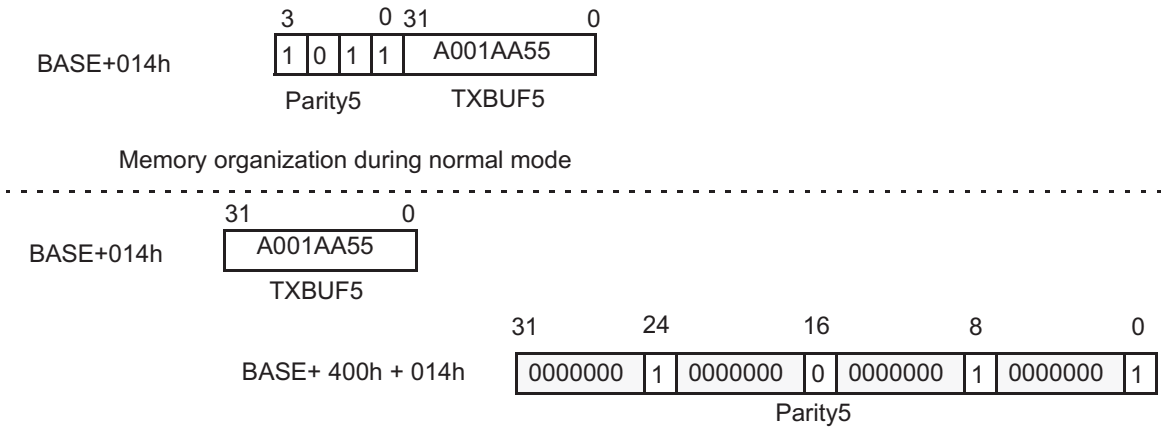


21.10.1 Example of Parity Memory Organization

Suppose TXBUF5 (6th location in TXRAM) in the multi-buffer RAM is written with a value of A001_AA55. If the polarity of the parity is set to odd, the corresponding parity location parity5 will get updated with equivalent parity of 1011 in its field.

During parity-memory test mode, these bits can be individually byte addressed. The return data will be a byte adjusted with actual parity bit in the LSB of the byte. If a word is read from the word-boundary address of parity locations, then each bit of the 4-bit parity is byte-adjusted and a 32-bit word is returned. 0s will be padded into the parity bits to get each byte. See Figure 21-67 for a diagram.

Figure 21-67. Example of Memory-Mapped Parity Locations During Test Mode



Parity memory locations during test mode (memory mapped)

1 Shaded areas indicate reads return 0, writes have no effect. These registers are not physically present.

NOTE: Read Access to Parity Memory Locations

Parity memory locations can be read even without entering into parity memory test mode. Their address remains as in memory test mode. It is only to enter parity-memory test mode to enable write access to the parity memory locations.

21.11 MibSPI Pin Timing Parameters

The pin timings of SPI can be classified based on its mode of operation. In each mode, different configurations like Phase & Polarity affect the pin timings.

The pin directions are based on the mode of operation.

Master mode SPI:

- SPICLK (SPI Clock) - Output
- SPISIMO (SPI Slave In Master Out) - Output
- $\overline{\text{SPICS}}$ (SPI Slave Chip Selects) - Output
- SPISOMI (SPI Slave Out Master In) - Input
- $\overline{\text{SPIENA}}$ (SPI slave ready Enable) - Input

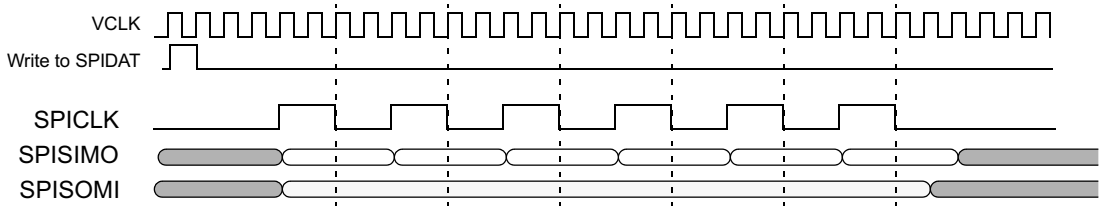
Slave mode SPI:

- SPICLK - Input
- SPISIMO - Input
- $\overline{\text{SPICS}}$ - Input
- SPISOMI - Output
- $\overline{\text{SPIENA}}$ - Output

NOTE: All the timing diagrams given below are with Phase = 0 and Polarity = 0, unless explicitly stated otherwise.

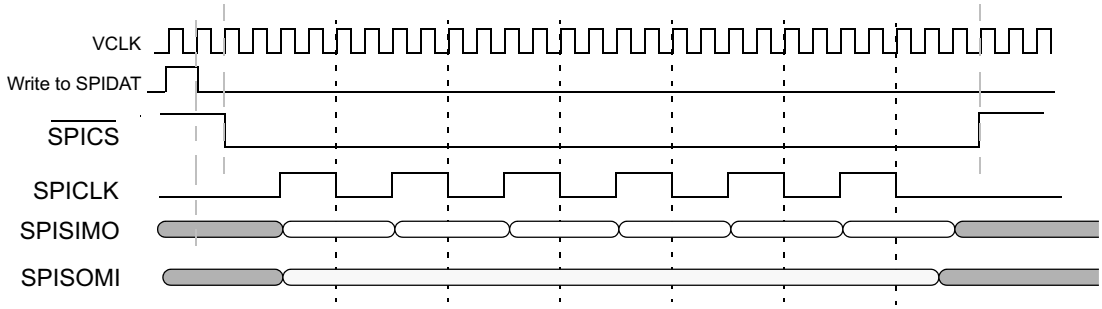
21.11.1 Master Mode Timings for SPI/MibSPI

Figure 21-68. SPI/MibSPI Pins During Master Mode 3-pin Configuration



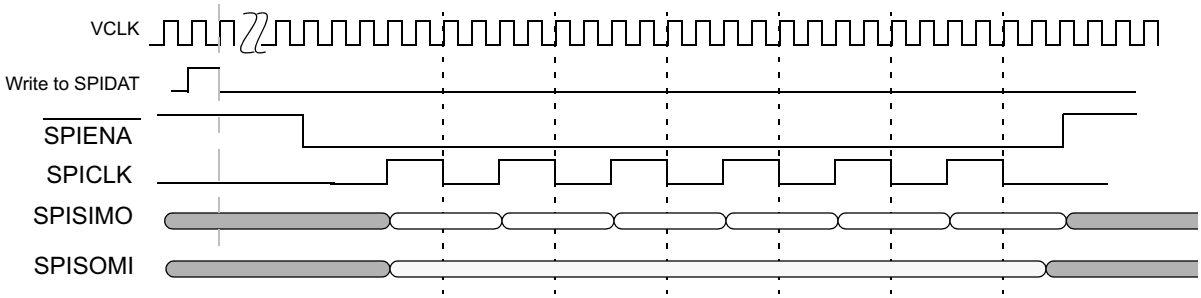
* Dotted vertical lines indicate the receive edges

Figure 21-69. SPI/MibSPI Pins During Master Mode 4-pin with $\overline{\text{SPICS}}$ Configuation



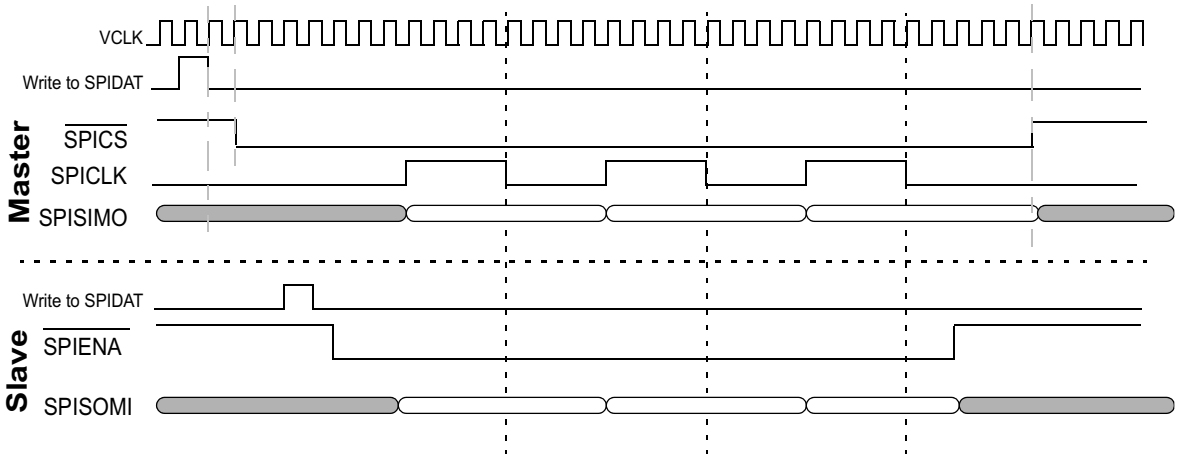
* Dotted vertical lines indicate the receive edges

Figure 21-70. SPI/MibSPI Pins During Master Mode in 4-pin with $\overline{\text{SPIENA}}$ Configuration



* De-activation of $\overline{\text{SPIENA}}$ pin is controlled by the Slave.
* Dotted vertical lines indicate the receive edges

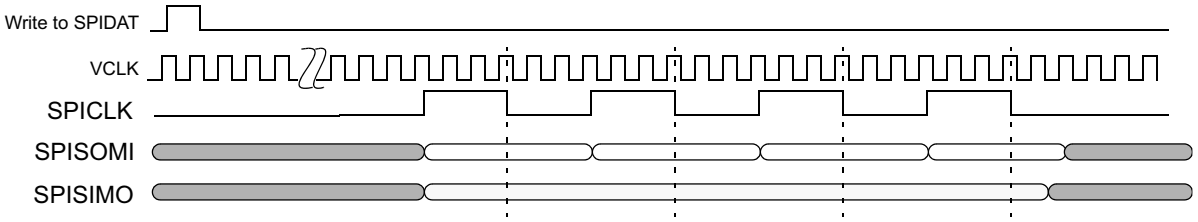
Figure 21-71. SPI/MibSPI Pins During Master/Slave Mode with 5-pin Configuration



* Dotted vertical lines indicate the receive edges for the Master
* ENABLE_HIGHZ is cleared to 0 in Slave SPI

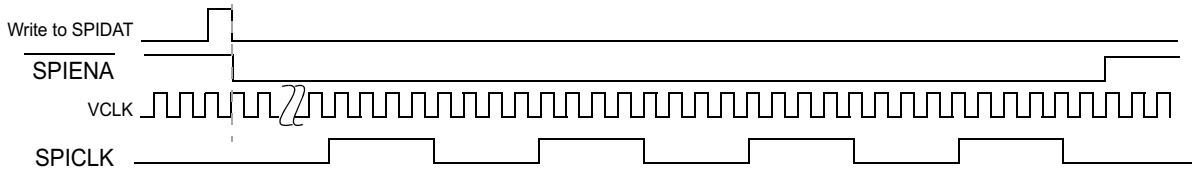
21.11.2 Slave Mode Timings for SPI/MibSPI

Figure 21-72. SPI/MibSPI Pins During Slave Mode 3-pin Configuration



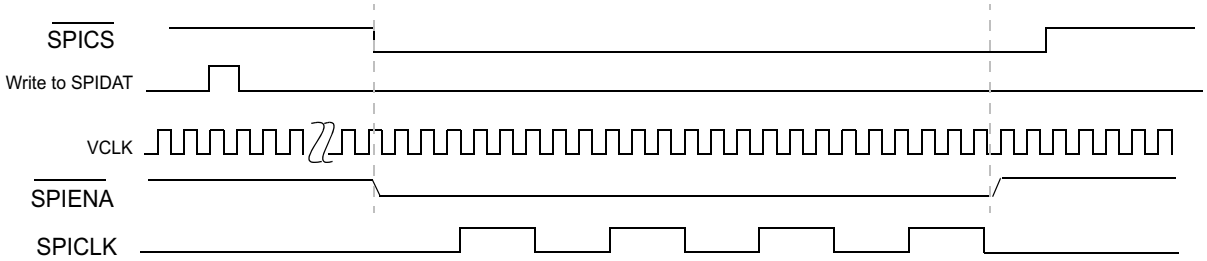
* Dotted vertical lines indicate the receive edges

Figure 21-73. SPI/MibSPI Pins During Slave Mode in 4-pin with $\overline{\text{SPIENA}}$ Configuration



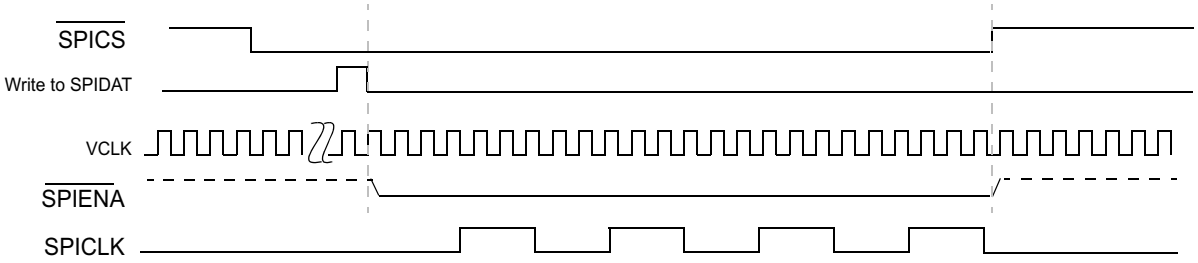
* Diagram shows a relationship between the $\overline{\text{SPIENA}}$ from Slave and SPICLK from Master

Figure 21-74. SPI/MibSPI Pins During Slave Mode in 5-pin Configuration - (Single Slave)



* ENABLE_HIGHZ is cleared to 0 in Slave SPI
* Diagram shows relationship between the $\overline{\text{SPICS}}$ from a Master to $\overline{\text{SPIENA}}$ from Slave SPI when $\overline{\text{SPIENA}}$ is configured in Push-Pull mode

Figure 21-75. SPI/MibSPI Pins During Slave Mode in 5-pin Configuration - (Single/Multi Slave)



* ENABLE_HIGHZ is set to 1 in Slave SPI
* Diagram shows relationship between the $\overline{\text{SPICS}}$ from a Master to $\overline{\text{SPIENA}}$ from Slave SPI when $\overline{\text{SPIENA}}$ is configured in High-Impedance mode

21.11.3 Master Mode Timing Parameter Details

In case of Master, the module drives out SPICLK. It also drives out the Transmit data on SPISIMO with respect to its internal SPICLK. In case of Master mode, the RX data on the SPISOMI pin is registered with respect to SPICLK received through the input buffer from the I/O pad.

If the chip select pin is functional, then the Master will drive out the $\overline{\text{SPICS}}$ pins before starting the SPICLK. If the $\overline{\text{SPIENA}}$ pin is functional, then the Master will wait for an active low from the Slave on the input pin to start the SPICLK.

21.11.4 Slave Mode Timing Parameter Details

In case of Slave mode, the module will drive only the SPISOMI and $\overline{\text{SPIENA}}$ pins. All other pins are inputs to it. The RX data on the SPISIMO pin will be registered with respect to the SPICLK pin. The Slave will use the $\overline{\text{SPICS}}$ pin to drive out the $\overline{\text{SPIENA}}$ pin if both are functional. If 4-pin with $\overline{\text{SPIENA}}$ is configured, then the Slave will drive out an active-low signal on the $\overline{\text{SPIENA}}$ pin when a new data is written to the TX Shift Register. Irrespective of 4-pin with $\overline{\text{SPIENA}}$ or 5-pin configuration, the Slave will deassert the $\overline{\text{SPIENA}}$ pin after the last bit is received. If ENABLE_HIGHZ (SPIINT0.24) bit is 0, the de-asserted value of the $\overline{\text{SPIENA}}$ pin will be 1. Otherwise, it will depend upon the internal pull up or pull down resistor (if implemented) depending upon the Specification of the Chip.

Serial Communication Interface (SCI)/Local Interconnect Network (LIN) Module

This chapter describes the serial communication interface (SCI) / local interconnect network (LIN) module. The SCI/LIN is compliant to the LIN 2.1 protocol specified in the *LIN Specification Package*. This module can be configured to operate in either SCI (UART) or LIN mode.

Topic **Page**

22.1 Introduction and Features

The SCI/LIN module can be programmed to work either as an SCI or as a LIN. The core of the module is an SCI. The SCI's hardware features are augmented to achieve LIN compatibility.

The SCI module is a universal asynchronous receiver-transmitter that implements the standard nonreturn to zero format. The SCI can be used to communicate, for example, through an RS-232 port or over a K-line.

The LIN standard is based on the SCI (UART) serial data link format. The communication concept is single-master/multiple-slave with a message identification for multi-cast transmission between any network nodes.

Throughout the chapter Compatibility Mode refers to SCI Mode functionary of SCI/LIN Module. The initial part of the chapter explains about the SCI functionality and later part about the LIN functionality. Though the register are common for LIN and SCI, the register descriptions has notes to identify the register / bit usage in different modes.

22.1.1 SCI Features

The following are the features of the SCI module:

- Standard universal asynchronous receiver-transmitter (UART) communication
- Supports full- or half-duplex operation
- Standard nonreturn to zero (NRZ) format
- Double-buffered receive and transmit functions in compatibility mode
- Supports two individually enabled interrupt lines: level 0 and level 1
- Configurable frame format of 3 to 13 bits per character based on the following:
  - Data word length programmable from one to eight bits
  - Additional address bit in address-bit mode
  - Parity programmable for zero or one parity bit, odd or even parity
  - Stop programmable for one or two stop bits
- Asynchronous or isosynchronous communication modes
- Two multiprocessor communication formats allow communication between more than two devices
- Sleep mode is available to free CPU resources during multiprocessor communication and then wake up to receive an incoming message
- The 24-bit programmable baud rate supports 2^{24} different baud rates provide high accuracy baud rate selection
- Five error flags and Seven status flags provide detailed information regarding SCI events
- Two external pins: LINRX and LINTX
- Multi-buffered receive and transmit units

NOTE: SCI/LIN module does not support UART hardware flow control. This feature can be implemented in software using a general purpose I/O pin.

22.1.2 LIN Features

The following are the features of the LIN module:

- Compatibility with LIN 1.3, 2.0 and 2.1protocols
- Configurable Baud Rate up to 20 Kbits/s
- Two external pins: LINRX and LINTX.
- Multi-buffered receive and transmit units
- Identification masks for message filtering
- Automatic master header generation
  - Programmable synchronization break field
  - Synchronization field
  - Identifier field
- Slave automatic synchronization
  - Synchronization break detection
  - Optional baud rate update
  - Synchronization validation
- 2^{31} programmable transmission rates with 7 fractional bits
- Wakeup on LINRX dominant level from transceiver
- Automatic wakeup support
  - Wakeup signal generation
  - Expiration times on wakeup signals
- Automatic bus idle detection
- Error detection
  - Bit error
  - Bus error
  - No-response error
  - Checksum error
  - Synchronization field error
  - Parity error
- 2 Interrupt lines with priority encoding for:
  - Receive
  - Transmit
  - ID, error, and status
- Support for LIN 2.0 checksum
- Enhanced synchronizer finite state machine (FSM) support for frame processing
- Enhanced handling of extended frames
- Enhanced baud rate generator
- Update wakeup/go to sleep

22.1.3 Block Diagram

The SCI/LIN module contains core SCI block with added sub-blocks to support LIN protocol.

Three Major components of the SCI Module are:

- Transmitter
- Baud Clock Generator
- Receiver

Transmitter (TX) contains two major registers to perform the double- buffering:

- The transmitter data buffer register (SCITD) contains data loaded by the CPU to be transferred to the shift register for transmission.
- The transmitter shift register (SCITXSHF) loads data from the data buffer (SCITD) and shifts data onto the LINTX pin, one bit at a time.

Baud Clock Generator

- A programmable baud generator produces either a baud clock scaled from VBUSP CLK.

Receiver (RX) contains two major registers to perform the double- buffering:

- The receiver shift register (SCIRXSHF) shifts data in from the LINRX pin one bit at a time and transfers completed data into the receive data buffer.
- The receiver data buffer register (SCIRD) contains received data transferred from the receiver shift register

The SCI receiver and transmitter are double-buffered, and each has its own separate enable and interrupt bits. The receiver and transmitter may each be operated independently or simultaneously in full duplex mode.

To ensure data integrity, the SCI checks the data it receives for breaks, parity, overrun, and framing errors. The bit rate (baud) is programmable to over 16 million different rates through a 24-bit baud-select register. Figure 22-1 shows the detailed SCI block diagram.

The SCI/LIN module is based on the standalone SCI with the addition of an error detector (parity calculator, checksum calculator, and bit monitor), a mask filter, a synchronizer, and a multi-buffered receiver and transmitter. The SCI interface and the baud generator are modified as part of the hardware enhancements for LIN compatibility. Figure 22-2 shows the SCI/LIN block diagram.

Figure 22-1. SCI Block Diagram

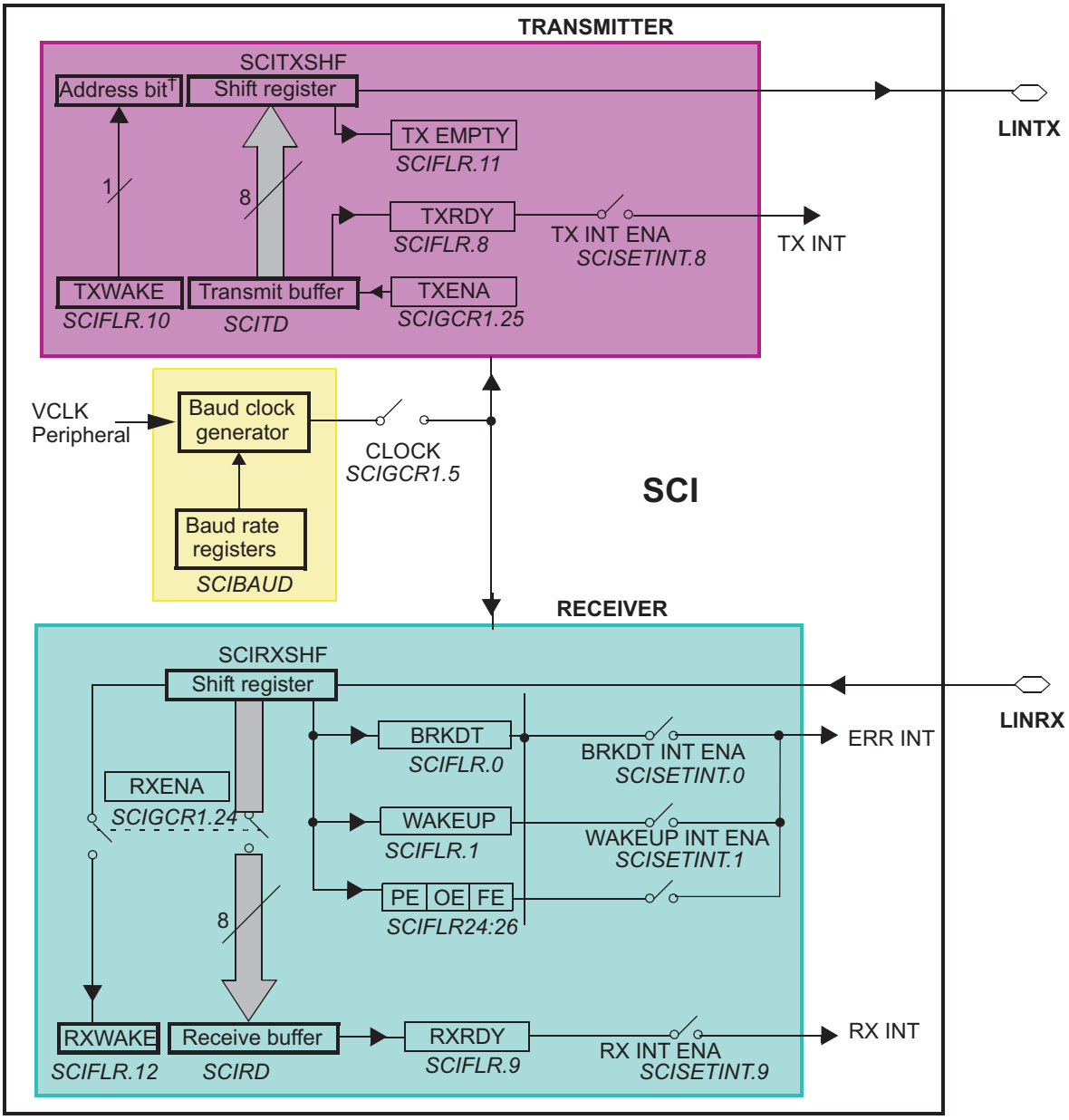
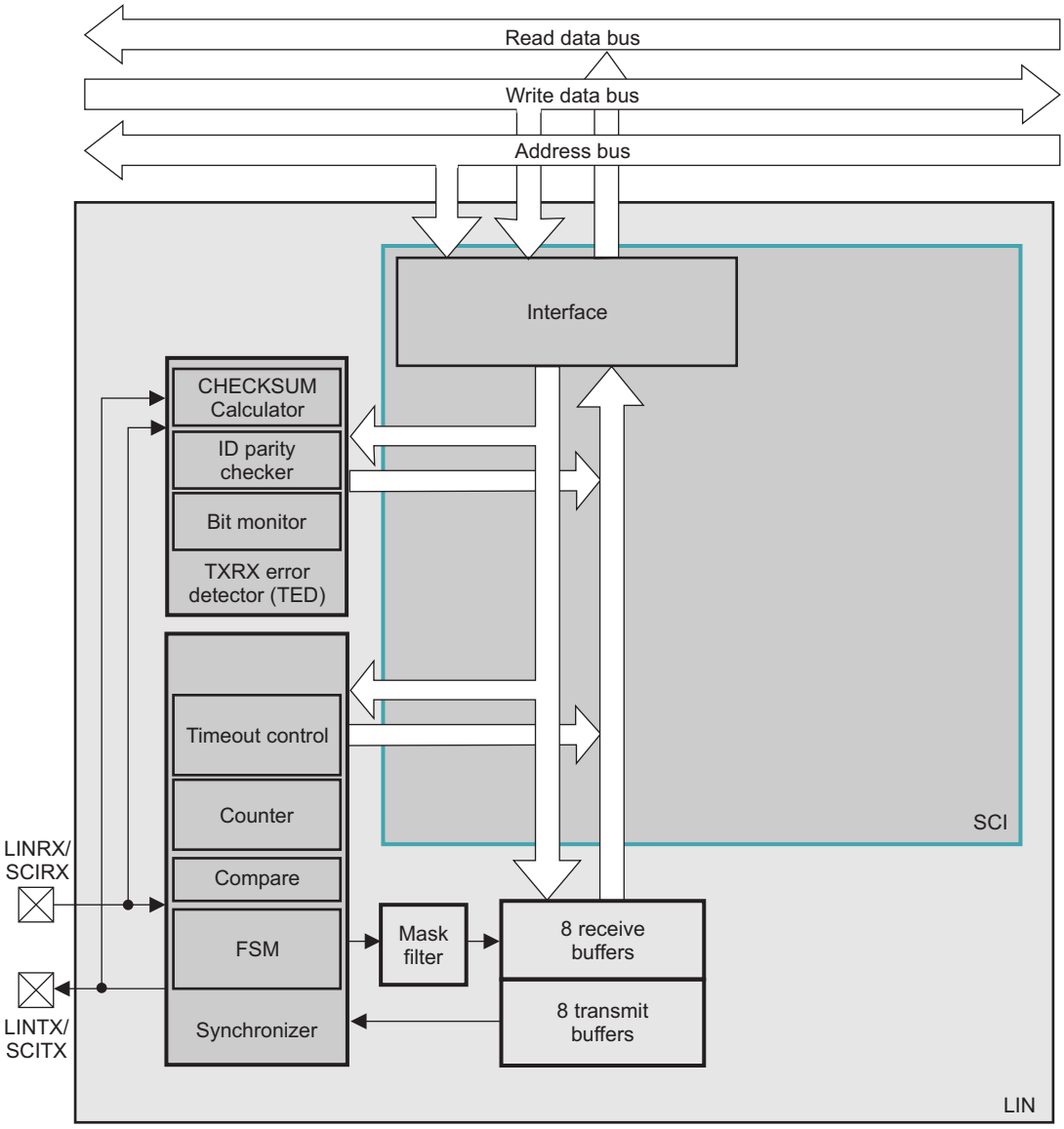


Figure 22-2. SCI/LIN Block Diagram



22.2 SCI Communication Formats

The SCI module can be configured to meet the requirements of many applications. Because communication formats vary depending on the specific application, many attributes of the SCI/LIN are user configurable. The list below describes these configuration options:

- SCI Frame format
- SCI Timing modes
- SCI Baud rate
- SCI Multiprocessor modes

22.2.1 SCI Frame Formats

The SCI uses a programmable frame format. All frames consist of the following:

- One start bit
- One to eight data bits
- Zero or one address bit
- Zero or one parity bit
- One or two stop bits

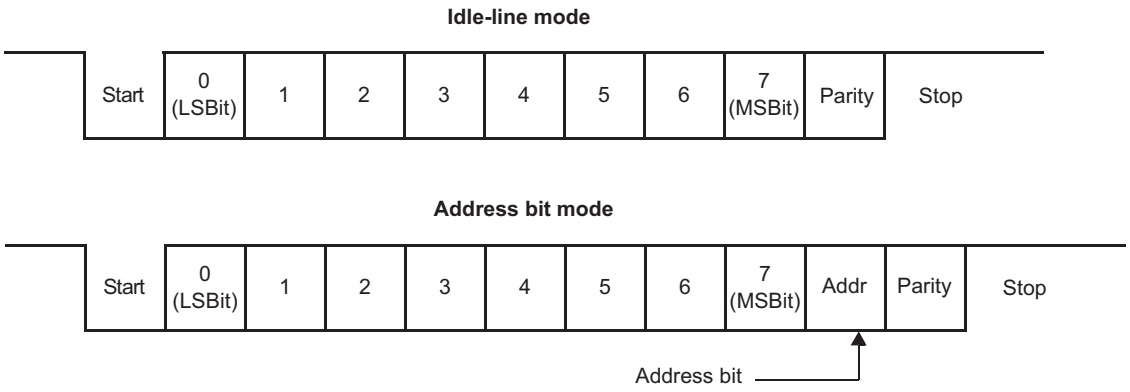
The frame format for both the transmitter and receiver is programmable through the bits in the SCIGCR1 register. Both receive and transmit data is in nonreturn to zero (NRZ) format, which means that the transmit and receive lines are at logic high when idle. Each frame transmission begins with a start bit, in which the transmitter pulls the SCI line low (logic low). Following the start bit, the frame data is sent and received least significant bit first (LSB).

An address bit is present in each frame if the SCI is configured to be in address-bit mode but is not present in any frame if the SCI is configured for idle-line mode. The format of frames with and without the address bit is illustrated in Figure 22-3.

A parity bit is present in every frame when the PARITY ENA bit is set. The value of the parity bit depends on the number of one bits in the frame and whether odd or even parity has been selected via the PARITY ENA bit. Both examples in Figure 22-3 have parity enabled.

All frames include one stop bit, which is always a high level. This high level at the end of each frame is used to indicate the end of a frame to ensure synchronization between communicating devices. Two stop bits are transmitted if the STOP bit in SCIGCR1 register is set. The examples shown in Figure 22-3 use one stop bit per frame.

Figure 22-3. Typical SCI Data Frame Formats



22.2.2 SCI Timing Mode

The SCI can be configured to use asynchronous or isosynchronous timing using TIMING MODE bit in SCIGCR1 register.

22.2.2.1 Asynchronous Timing Mode

The asynchronous timing mode uses only the receive and transmit data lines to interface with devices using the standard universal asynchronous receiver- transmitter (UART) protocol.

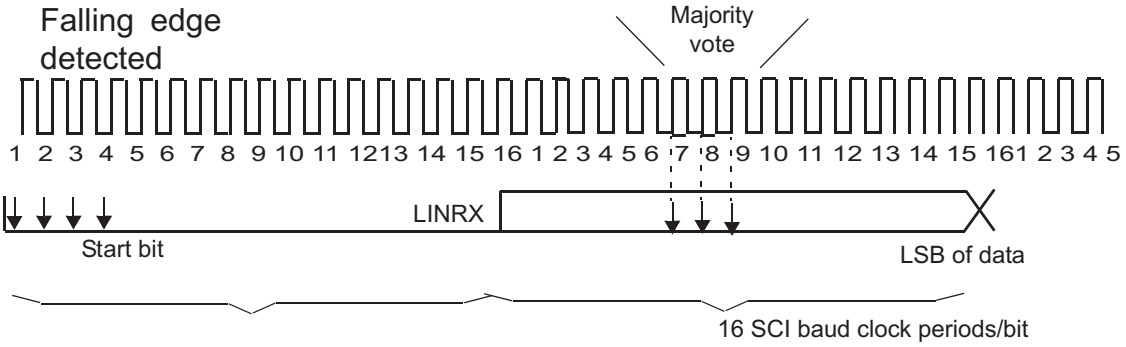
In the asynchronous timing mode, each bit in a frame has a duration of 16 SCI baud clock periods. Each bit therefore consists of 16 samples (one for each clock period). When the SCI is using asynchronous mode, the baud rates of all communicating devices must match as closely as possible. Receive errors result from devices communicating at different baud rates.

With the receiver in the asynchronous timing mode, the SCI detects a valid start bit if the first four samples after a falling edge on the LINRX pin are of logic level 0. As soon as a falling edge is detected on LINRX, the SCI assumes that a frame is being received and synchronizes itself to the bus.

To prevent interpreting noise as Start bit SCI expects LINRX line to be low for at least four contiguous SCI baud clock periods to detect a valid start bit. The bus is considered idle if this condition is not met. When a valid start bit is detected, the SCI determines the value of each bit by sampling the LINRX line value during the seventh, eighth, and ninth SCI baud clock periods. A majority vote of these three samples is used to determine the value stored in the SCI receiver shift register. By sampling in the middle of the bit, the SCI reduces errors caused by propagation delays and rise and fall times and data line noises. Figure 22-4 illustrates how the receiver samples a start bit and a data bit in asynchronous timing mode.

The transmitter transmits each bit for a duration of 16 SCI baud clock periods. During the first clock period for a bit, the transmitter shifts the value of that bit onto the LINTX pin. The transmitter then holds the current bit value on LINTX for 16 SCI baud clock periods.

Figure 22-4. Asynchronous Communication Bit Timing



22.2.2.2 Isosynchronous Timing Mode

In isosynchronous timing mode, each bit in a frame has a duration of exactly 1 baud clock period and therefore consists of a single sample. With this timing configuration, the transmitter and receiver are required to make use of the SCICLK pin to synchronize communication with other SCI. **This mode is not supported on this device because SCICLK pin is not available.**

22.2.3 SCI Baud Rate

The SCI/LIN has an internally generated serial clock determined by the peripheral VCLK and the prescalers P and M in this register. The SCI uses the 24-bit integer prescaler P value in the BRS register to select the required baud rates. The additional 4-bit fractional divider M refines the baud rate selection.

In asynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$SCICLK\ Frequency = \frac{VCLK\ Frequency}{P + 1 + \frac{M}{16}}$$

$$Asynchronous\ baud\ value = \frac{SCICLK\ Frequency}{16}$$

For P = 0,

$$Asynchronous\ baud\ value = \frac{VCLK\ Frequency}{32} \quad (40)$$

22.2.3.1 Superfractional Divider, SCI Asynchronous Mode

The superfractional divider is available in SCI asynchronous mode (idle-line and address-bit mode). Building on the 4-bit fractional divider M (BRS[27:24]), the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 22-2. The bits with a 1 in the table will have an additional VCLK period added to their T_{bit}. If the character length is more than 10, then the modulation table will be a rolled-over version of the original table (Table 22-1), as shown in Table 22-2.

The baud rate will vary over a data field to average according to the BRS[30:28] value by a "**d**" fraction of the peripheral internal clock: **0<d<1.** Refer Figure 22-5 for a simple Average "d' calculation based on "U" value (BRS[30:28]).

The instantaneous bit time is expressed in terms of T_{VCLK} as follows:

For all P other than 0, and all M and d (0 or 1),

$$T^{i}bit = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK} \quad (41)$$

For P = 0 $T_{bit} = 32T_{VCLK}$

The averaged bit time is expressed in terms of T_{VCLK} as follows:

For all P other than 0, and all M and d (0<d<1),

$$T^{a}bit = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK} \quad (42)$$

For P = 0 $T_{bit} = 32T_{VCLK}$

Table 22-1. Superfractional Bit Modulation for SCI Mode (Normal Configuration)(1)

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0h	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0
2h	1	0	0	0	1	0	0	0	1	0
3h	1	0	1	0	1	0	0	0	1	0
4h	1	0	1	0	1	0	1	0	1	0
5h	1	1	1	0	1	0	1	0	1	1
6h	1	1	1	0	1	1	1	0	1	1
7h	1	1	1	1	1	1	1	0	1	1

(1) Normal configuration = Start + 8 Data Bits + Stop Bit

Table 22-2. Superfractional Bit Modulation for SCI Mode (Maximum Configuration)(1)

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Addr	Parity	Stop0	Stop1
0h	0	0	0	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0	0	0	0
2h	1	0	0	0	1	0	0	0	1	0	0	0	1
3h	1	0	1	0	1	0	0	0	1	0	1	0	1
4h	1	0	1	0	1	0	1	0	1	0	1	0	1
5h	1	1	1	0	1	0	1	0	1	1	1	0	1
6h	1	1	1	0	1	1	1	0	1	1	1	0	1
7h	1	1	1	1	1	1	1	0	1	1	1	1	1

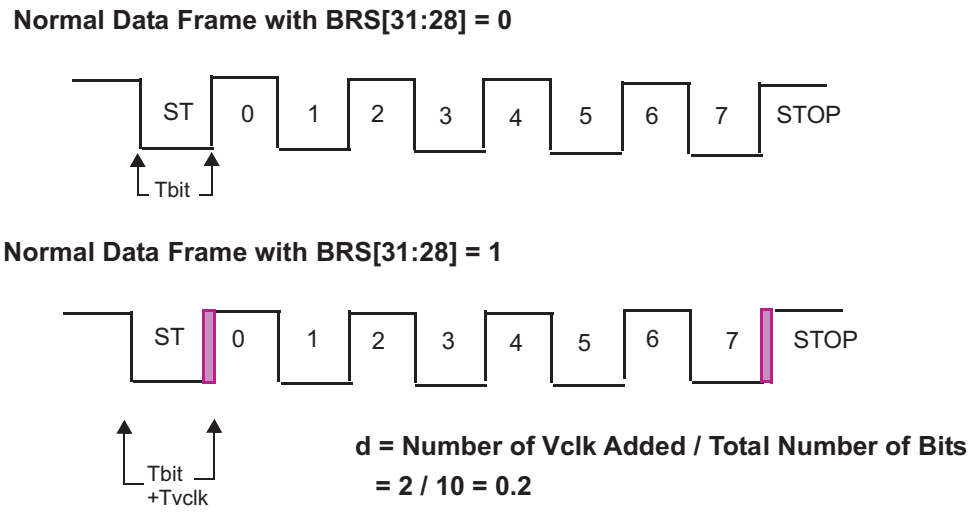
(1) Maximum configuration = Start + 8 Data Bits + Addr Bit + Parity Bit + Stop Bit 0 + Stop Bit 1

Table 22-3. SCI Mode (Minimum Configuration)(1)

BRS[30:28]	Start Bit	D[0]	Stop
0h	0	0	0
1h	1	0	0
2h	1	0	0
3h	1	0	1
4h	1	0	1
5h	1	1	1
6h	1	1	1
7h	1	1	1

(1) Minimum configuration = Start + 1 Data Bits + Stop Bit

Figure 22-5. Superfractional Divider Example



22.2.4 SCI Multiprocessor Communication Modes

In some applications, the SCI may be connected to more than one serial communication device. In such a multiprocessor configuration, several frames of data may be sent to all connected devices or to an individual device. In the case of data sent to an individual device, the receiving devices must determine when they are being addressed. When a message is not intended for them, the devices can ignore the following data. When only two devices make up the SCI network, addressing is not needed, so multiprocessor communication schemes are not required.

SCI supports two multiprocessor communication modes which can be selected using COMM MODE bit:

- Idle-Line Mode
- Address Bit Mode

When the SCI is not used in a multiprocessor environment, software can consider all frames as data frames. In this case, the only distinction between the idle-line and address-bit modes is the presence of an extra bit (the address bit) in each frame sent with the address-bit protocol.

The SCI allows full-duplex communication where data can be sent and received via the transmit and receive pins simultaneously. However, the protocol used by the SCI assumes that only one device transmits data on the same bus line at any one time. No arbitration is done by the SCI.

22.2.4.1 Idle-Line Multiprocessor Modes

In idle-line multiprocessor mode, a frame that is preceded by an idle period (10 or more idle bits) is an address frame. A frame that is preceded by fewer than 10 idle bits is a data frame. Figure 22-6 illustrates the format of several blocks and frames with idle-line mode.

There are two ways to transmit an address frame using idle-line mode:

Method 1: In software, deliberately leave an idle period between the transmission of the last data frame of the previous block and the address frame of the new block.

Method 2: Configure the SCI to automatically send an idle period between the last data frame of the previous block and the address frame of the new block.

Although Method 1 is only accomplished by a delay loop in software, Method 2 can be implemented by using the transmit buffer and the TXWAKE bit in the following manner:

Step 1: Write a 1 to the TXWAKE bit.

Step 2: Write a dummy data value to the SCITD register. This triggers the SCI to begin the idle period as soon as the transmitter shift register is empty.

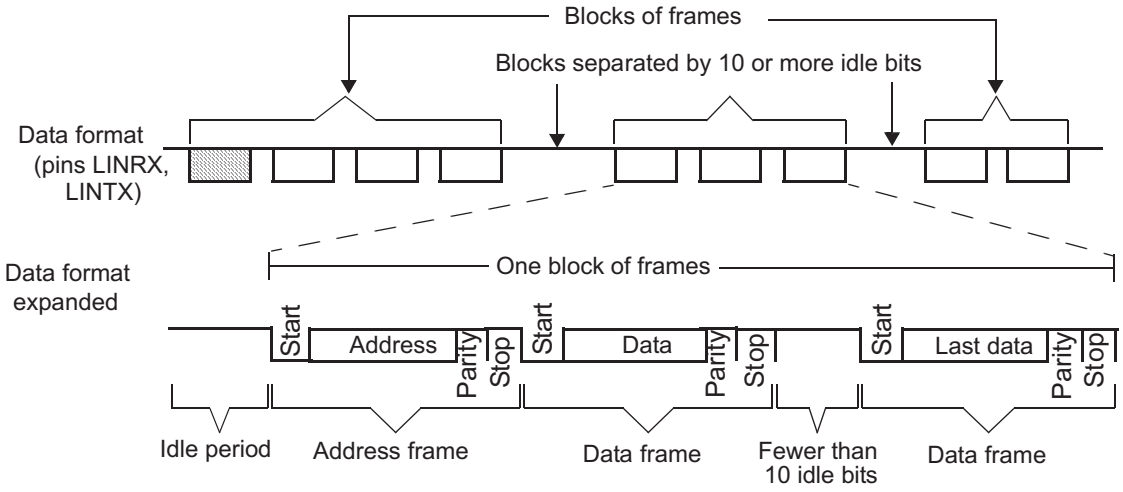
Step 3: Wait for the SCI to clear the TXWAKE flag.

Step 4: Write the address value to SCITD.

As indicated by Step 3, software should wait for the SCI to clear the TXWAKE bit. However, the SCI clears the TXWAKE bit at the same time it sets TXRDY (that is, transfers data from SCITD into SCITXSHF). Therefore, if the TX INT ENA bit is set, the transfer of data from SCITD to SCITXSHF causes an interrupt to be generated at the same time that the SCI clears the TXWAKE bit. If this interrupt method is used, software is not required to poll the TXWAKE bit waiting for the SCI to clear it.

When idle-line multiprocessor communications are used, software must ensure that the idle time exceeds 10 bit periods before addresses (using one of the methods mentioned above), and software must also ensure that data frames are written to the transmitter quickly enough to be sent without a delay of 10 bit periods between frames. Failure to comply with these conditions will result in data interpretation errors by other devices receiving the transmission.

Figure 22-6. Idle-Line Multiprocessor Communication Format



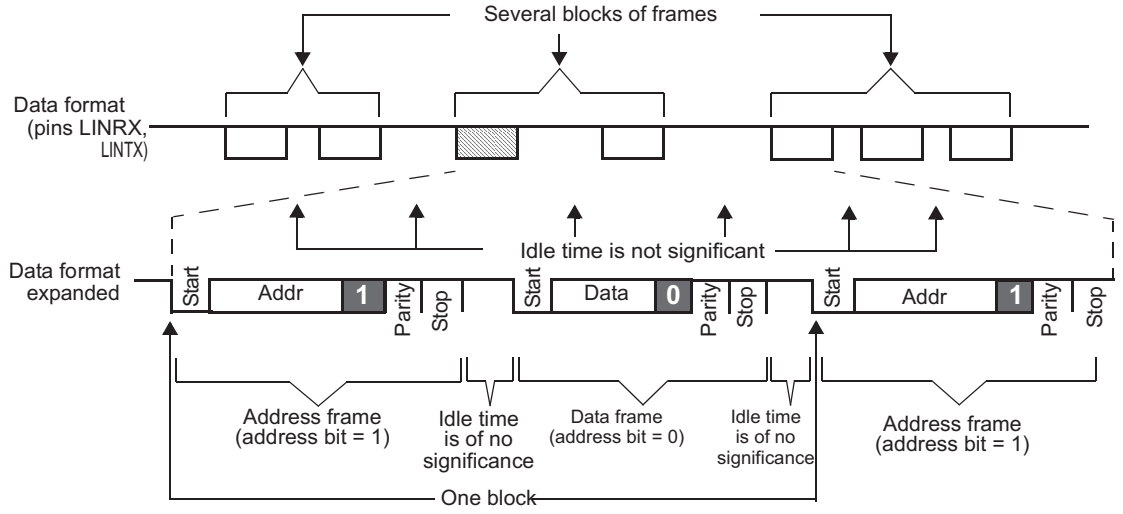
22.2.4.2 Address-Bit Multiprocessor Mode

In the address-bit protocol, each frame has an extra bit immediately following the data field called an address bit. A frame with the address bit set to 1 is an address frame; a frame with the address bit set to 0 is a data frame. The idle period timing is irrelevant in this mode. Figure 22-7 illustrates the format of several blocks and frames with the address-bit mode.

When address-bit mode is used, the value of the TXWAKE bit is the value sent as the address bit. To send an address frame, software must set the TXWAKE bit. This bit is cleared as the contents of the SCITD are shifted from the TXWAKE register so that all frames sent are data except when the TXWAKE bit is written as a 1.

No dummy write to SCITD is required before an address frame is sent in address-bit mode. The first byte written to SCITD after the TXWAKE bit is written to 1 is transmitted with the address bit set when address-bit mode is used.

Figure 22-7. Address-Bit Multiprocessor Communication Format



22.2.5 SCI Multi-Buffered Mode

To reduce CPU load when Receiving or Transmitting data in interrupt mode, the SCI/LIN module has eight separate receive and transmit buffers. Multi-buffered mode is enabled by setting the MBUF MODE bit in the SCIGCR1 register.

The multi-buffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers and TDy transmit buffers register to SCITXSHF register. The 3-bit compare register contains the number of data bytes expected to be received or transmitted. the LENGTH value in SCIFORMAT register indicates the expected length and is used to load the 3-bit compare register.

A receive interrupt (RX interrupt; see the SCIINTVECT0 and SCIINTVECT1registers) and a receive ready RXRDY flag set in SCIFLR register could occur after receiving a response if there are no response receive errors for the frame (such as a frame error or overrun error).

A transmit interrupt (TX interrupt) and a transmit ready flag (TXRDY flag in SCIFLR register could occur after transmitting a response.

Figure 22-8 and Figure 22-9 shows the receive and transmit multi-buffer functional block diagram.

Figure 22-8. Receive Buffers

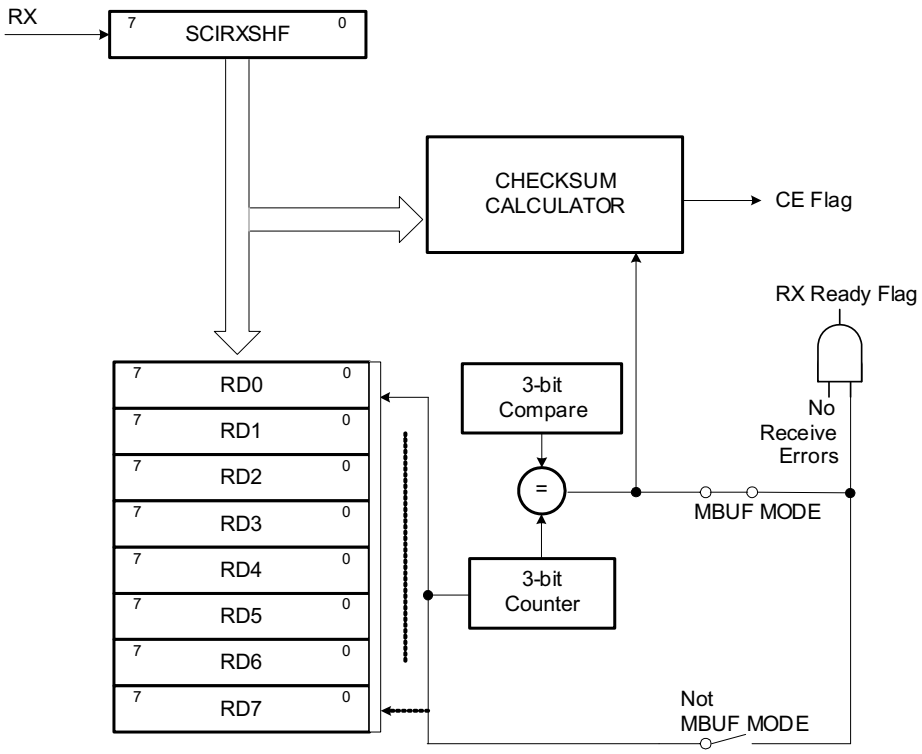
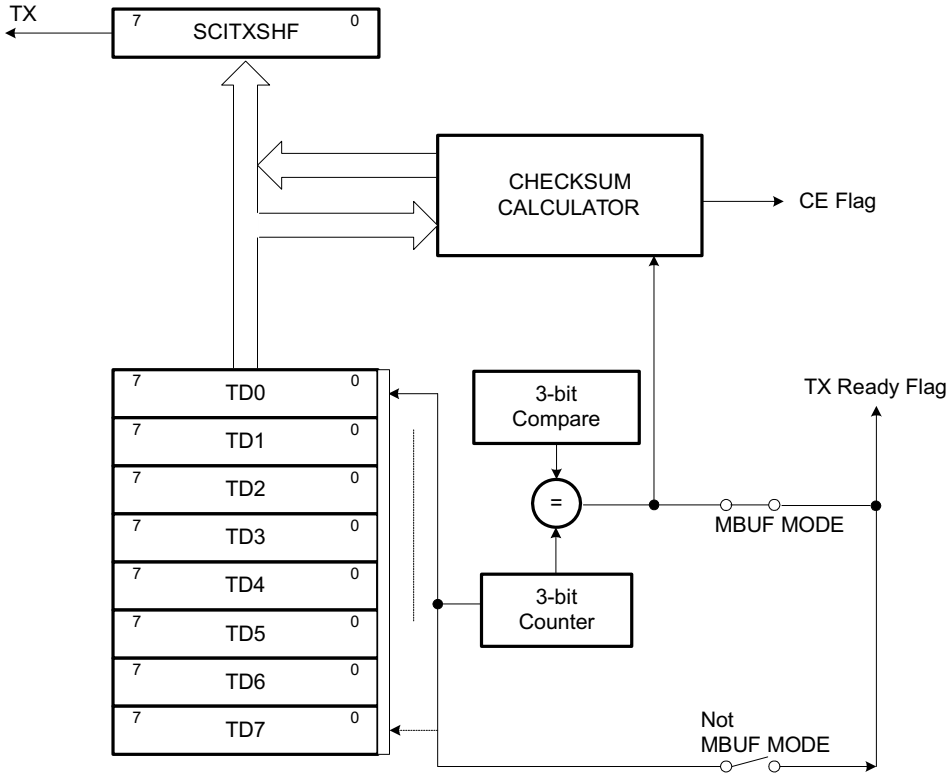


Figure 22-9. Transmit Buffers



22.3 SCI Interrupts

The SCI/LIN module has two interrupt lines, level 0 and level 1, to the vectored interrupt manager (VIM) module (see Figure 22-10). Two offset registers SCIINTVECT0 and SCIINTVECT1 determine which flag triggered the interrupt according to the respective priority encoders. Each interrupt condition has a bit to enable/disable the interrupt in the SCISETINT and SCICLRINT registers respectively.

Each interrupt also has a bit that can be set as interrupt level 0(INT0) or as interrupt level 1(INT1). By default, interrupts are in interrupt level 0. SCISETINTLVL sets a given interrupt to level1. SCICLEARINTLVL resets a given interrupt level to the default level 0.

The interrupt vector registers SCIINTVECT0 and SCIINTVECT1 return the vector of the pending interrupt line INT0 or INT1. If more than one interrupt is pending, the interrupt vector register holds the highest priority interrupt.

Figure 22-10. General Interrupt Scheme

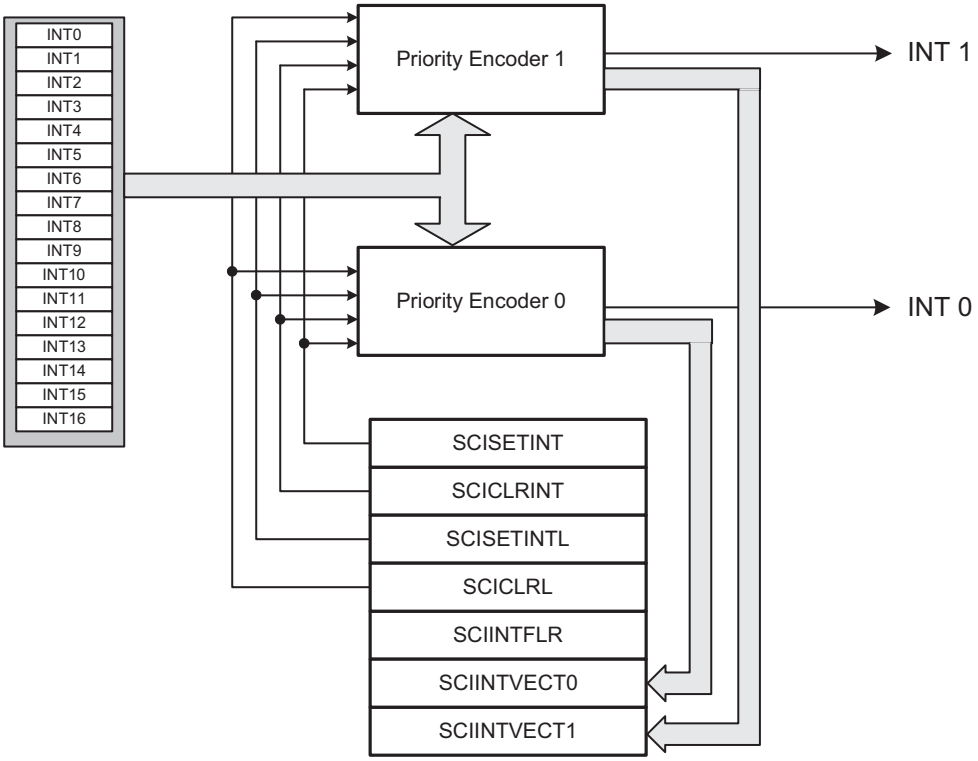
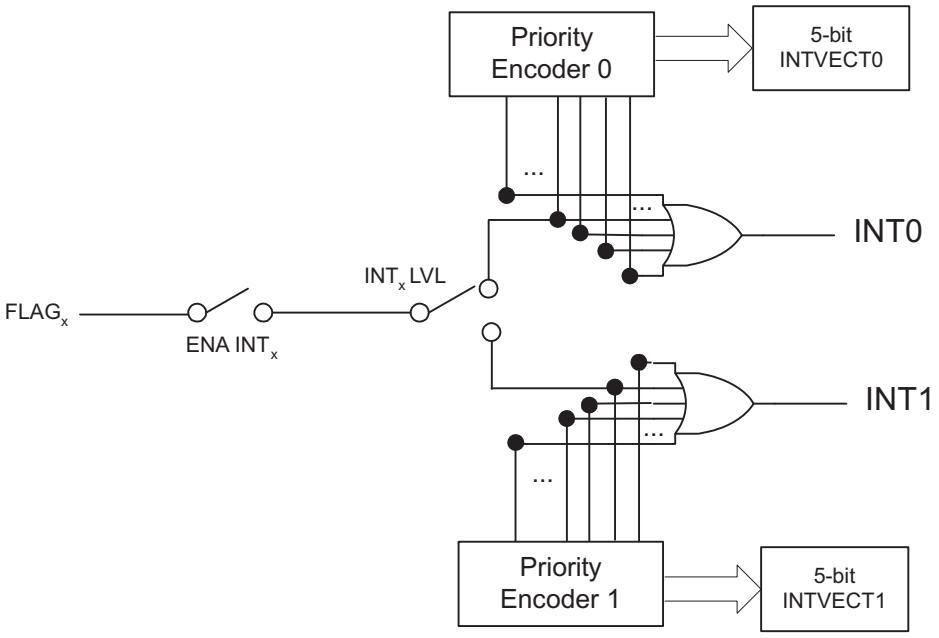


Figure 22-11. Interrupt Generation for Given Flags



22.3.1 Transmit Interrupt

To use transmit interrupt functionality, SET TX INT bit must be enabled. The transmit ready (TXRDY) flag is set when the SCI transfers the contents of SCITD to the shift register, SCITXSHF. The TXRDY flag indicates that SCITD is ready to be loaded with more data. In addition, the SCI sets the TX EMPTY bit if both the SCITD and SCITXSHF registers are empty. If the SET TX INT bit is set, then a transmit interrupt is generated when the TXRDY flag goes high. Transmit Interrupt is not generated immediately after setting the SET TX INT bit. Transmit Interrupt is generated only after the first transfer from SCITD to SCITXSHF. That is, the first data has to be written to SCITD by the User before any interrupt gets generated. To transmit further data the user can write data to SCITD in the transmit Interrupt service routine.

Writing data to the SCITD register clears the TXRDY bit. When this data has been moved to the SCITXSHF register, the TXRDY bit is set again. The interrupt request can be suspended by setting the CLR TX INT bit; however, when the SET TX INT bit is again set to 1, the TXRDY interrupt is asserted again. The transmit interrupt request can be eliminated until the next series of values is written to SCITD, by disabling the transmitter via the TXENA bit, by a software reset SWnRST, or by a device hardware reset.

22.3.2 Receive Interrupt

The receive ready (RXRDY) flag is set when the SCI transfers newly received data from SCIRXSHF to SCIRD. The RXRDY flag therefore indicates that the SCI has new data to be read. Receive interrupts are enabled by the SET RX INT bit. If the SET RX INT is set when the SCI sets the RXRDY flag, then a receive interrupt is generated. The received data can be read in the Interrupt Service routine.

22.3.3 WakeUp Interrupt

SCI sets the WAKEUP flag if bus activity on the RX line either prevents power-down mode from being entered, or RX line activity causes an exit from power-down mode. If enabled (SET WAKEUP INT), wakeup interrupt is triggered once WAKEUP flag is set.

22.3.4 Error Interrupts

The following error detection are supported with Interrupt by the SCI module:

- Parity errors (PE)
- Frame errors (FE)
- Break Detect errors (BRKDT)
- Overrun errors (OE)
- Bit errors (BE)

If all of these errors (PE, FE, BRKDT, OE, BE) are flagged, an interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame. Each of these flags is located in the receiver status (SCIFLR) register.

There are 16 interrupt sources in the SCI/LIN module, In SCI mode 8 interrupts are supported, as seen in Table 22-4.

Table 22-4. SCI/LIN Interrupts

Offset [1]	Interrupt	Applicable to SCI	Applicable to LIN
0	No interrupt		
1	Wakeup	Yes	Yes
2	Inconsistent-synch-field error	No	Yes
3	Parity error	Yes	Yes
4	ID	No	Yes
5	Physical bus error	No	Yes
6	Frame error	Yes	Yes
7	Break detect	Yes	No
8	Checksum error	No	Yes
9	Overrun error	Yes	Yes
10	Bit error	Yes	Yes
11	Receive	Yes	Yes
12	Transmit	Yes	Yes
13	No-response error	No	Yes
14	Timeout after wakeup signal (150 ms)	No	Yes
15	Timeout after three wakeup signals (1.5 s)	No	Yes
16	Timeout (Bus Idle, 4s)	No	Yes

(1) Offset 1 is the highest priority. Offset 16 is the lowest priority.

22.4 SCI Configurations

Before the SCI sends or receives data, its registers should be properly configured. Upon power-up or a system-level reset, each bit in the SCI registers is set to a default state. The registers are writable only after the RESET bit in the SCIGCR0 register is set to 1. Of particular importance is the SWnRST bit in the SCIGCR1 register. The SWnRST is an active-low bit initialized to 0 and keeps the SCI in a reset state until it is programmed to 1. Therefore, all SCI configuration should be completed before a 1 is written to the SWnRST bit.

The following list details the configuration steps that software should perform prior to the transmission or reception of data. As long as the SWnRST bit is cleared to 0 the entire time that the SCI is being configured, the order in which the registers are programmed is not important.

- Enable SCI by setting the RESET bit to 1.
- Clear the SWnRST bit to 0 before SCI is configured.
- Select the desired frame format by programming the SCIGCR1 register.
- Set both the RX FUNC and TX FUNC bits in SCIPIO0 to 1 to configure the LINRX and LINTX pins for SCI functionality.
- Select the baud rate to be used for communication by programming the BRS register.
- Set the CLOCK bit in SCIGCR1 to 1 to select the internal clock.
- Set the CONT bit in SCIGCR1 to 1 to make SCI not halt for an emulation breakpoint until its current reception or transmission is complete (this bit is used only in an emulation environment).
- Set the LOOP BACK bit in SCIGCR1 to 1 to connect the transmitter to the receiver internally (this feature is used to perform a self-test).
- Set the RXENA bit in SCIGCR1 to 1, if data is to be received.
- Set the TXENA bitin SCIGCR1 to 1, if data is to be transmitted.
- Set the SWnRST bit to 1 after SCI is configured.
- Perform receiving or transmitting data (see Section 22.4.1 or Section 22.4.2).

22.4.1 Receiving Data

The SCI receiver is enabled to receive messages if both the RX FUNC bit and the RXENA bit are set to 1. If the RX FUNC bit is not set, the LINRX pin functions as a general-purpose I/O pin rather than as an SCI function pin.

SCI module can receive data in one of the following modes:

- Single-Buffer (Normal) Mode
- Multi-Buffer Mode

After a valid idle period is detected, data is automatically received as it arrives on the LINRX pin.

22.4.1.1 Receiving Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is cleared to 0. In this mode, SCI sets the RXRDY bit when it transfers newly received data from SCIRXSHF to SCIRD. The SCI clears the RXRDY bit after the new data in SCIRD has been read. Also, as data is transferred from SCIRXSHF to SCIRD, the SCI sets the FE, OE, or PE flags if any of these error conditions were detected in the received data. These error conditions are supported with configurable interrupt capability. The wake-up and break-detect status bits are also set if one of these errors occurs, but they do not necessarily occur at the same time that new data is being loaded into SCIRD.

You can receive data by:

1. Polling Receive Ready Flag
2. Receive Interrupt

In polling method, software can poll for the RXRDY bit and read the data from the SCIRD register once the RXRDY bit is set high. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use the interrupt method. To use the interrupt method, the SET RX INT bit is set. An interrupt is generated the moment the RXRDY bit is set.

22.4.1.2 Receiving Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is set to 1. In this mode, SCI sets the RXRDY bit after receiving the programmed number of data in the receive buffer, the complete frame. The error condition detection logic is similar to the single-buffer mode, except that it monitors for the complete frame. Like single-buffer mode, you can use the polling or interrupt method to read the data. The SCI clears the RXRDY bit after the new data in SCIRD has been read.

22.4.2 Transmitting Data

The SCI transmitter is enabled if both the TX FUNC bit and the TXENA bit are set to 1. If the TX FUNC bit is not set, the LINTX pin functions as a general-purpose I/O pin rather than as an SCI function pin. Any value written to the SCITD before TXENA is set to 1 is not transmitted. Both of these control bits allow for the SCI transmitter to be held inactive independently of the receiver.

SCI module can transmit data in one of the following modes:

- Single-Buffer (Normal) Mode
- Multi-Buffered or Buffered SCI Mode.

22.4.2.1 Transmitting Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is cleared to 0. In this mode, SCI waits for data to be written to SCITD, transfers it to SCITXSHF, and transmits the data. The TXRDY and TX EMPTY bits indicate the status of the transmit buffers. That is, when the transmitter is ready for data to be written to SCITD, the TXRDY bit is set. Additionally, if both SCITD and SCITXSHF are empty, then the TX EMPTY bit is also set.

You can transmit data by:

1. Polling Transmit Ready Flag
2. Transmit Interrupt

In polling method, software can poll for the TXRDY bit to go high before writing the data to the SCITD register. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use the interrupt method. To use the interrupt method, the SET TX INT bit is set. An interrupt is generated the moment the TXRDY bit is set. When the SCI has completed transmission of all pending frames, the SCITXSHF register and SCITD are empty, the TXRDY bit is set, and an interrupt is generated, if enabled. Because all data has been transmitted, the interrupt should be halted. This can either be done by disabling the transmit interrupt (CLR TX INT) or by disabling the transmitter (clear TXENA bit).

NOTE: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

22.4.2.2 Transmitting Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is set to 1. Like single-buffer mode, you can use the polling or interrupt method to write the data to be transmitted. The transmitted data has to be written to the SCITD registers. SCI waits for data to be written to the SCITD register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically.

22.5 SCI Low-Power Mode

The SCI/LIN can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SCI/LIN. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and the module registers. Setting the POWERDOWN bit causes the SCI to enter local low-power mode and clearing the POWERDOWN bit causes SCI/LIN to exit from local low-power mode. All the registers are accessible during local power-down mode as any register access enables the clock to SCI for that particular access alone.

The wake-up interrupt is used to allow the SCI to exit low-power mode automatically when a low level is detected on the LINRX pin and also this clears the POWERDOWN bit. If wake-up interrupt is disabled, then the SCI/LIN immediately enters low-power mode whenever it is requested and also any activity on the LINRX pin does not cause the SCI to exit low-power mode.

NOTE: Enabling Local Low-Power Mode During Receive and Transmit

If the wake-up interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI immediately generates a wake-up interrupt to clear the powerdown bit and prevents the SCI from entering low-power mode and thus completes the current reception. Otherwise, if the wake-up interrupt is disabled, then the SCI completes the current reception and then enters the low-power mode.

22.5.1 Sleep Mode for Multiprocessor Communication

When the SCI receives data and transfers that data from SCIRXSHF to SCIRD, the RXRDY bit is set and if RX INT ENA is set, the SCI also generates an interrupt. The interrupt triggers the CPU to read the newly received frame before another one is received. In multiprocessor communication modes, this default behavior may be enhanced to provide selective indication of new data. When SCI receives an address frame that does not match its address, the device can ignore the data following this non-matching address until the next address frame by using sleep mode. Sleep mode can be used with both idle-line and address-bit multiprocessor modes.

If sleep mode is enabled by the SLEEP bit, then the SCI transfers data from SCIRXSHF to SCIRD only for address frames. Therefore, in sleep mode, all data frames are assembled in the SCIRXSHF register without being shifted into the SCIRD and without initiating a receive interrupt. Upon reception of an address frame, the contents of the SCIRXSHF are moved into SCIRD, and the software must read SCIRD and determine if the SCI is being addressed by comparing the received address against the address previously set in the software and stored somewhere in memory (the SCI does not have hardware available for address comparison). If the SCI is being addressed, the software must clear the SLEEP bit so that the SCI will load SCIRD with the data of the data frames that follow the address frame.

When the SCI has been addressed and sleep mode has been disabled (in software) to allow the receipt of data, the SCI should check the RXWAKE bit (SCIFLR.12) to determine when the next address has been received. This bit is set to 1 if the current value in SCIRD is an address and set to 0 if SCIRD contains data. If the RXWAKE bit is set, then software should check the address in SCIRD against its own address. If it is still being addressed, then sleep mode should remain disabled. Otherwise, the SLEEP bit should be set again.

Following is a sequence of events typical of sleep mode operation:

- The SCI is configured and both sleep mode and receive actions are enabled.
- An address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is not being addressed, so the value of the SLEEP bit is not changed.
- Several data frames are shifted into SCIRXSHF, but no data is moved to SCIRD and no receive interrupts are generated.
- A new address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is being addressed and clears the SLEEP bit.
- Data shifted into SCIRXSHF is transferred to SCIRD, and a receive interrupt is generated after each data frame is received.
- In each interrupt routine, software checks RXWAKE to determine if the current frame is an address frame.
- Another address frame is received, RXWAKE is set, software determines that the SCI is not being addressed and sets the SLEEP bit back to 1. No receive interrupts are generated for the data frames following this address frame.

By ignoring data frames that are not intended for the device, fewer interrupts are generated. These interrupts would otherwise require CPU intervention to read data that is of no significance to this specific device. Using sleep mode can help free some CPU resources.

Except for the RXRDY flag, the SCI continues to update the receiver status flags (see Table 22-12) while sleep mode is active. In this way, if an error occurs on the receive line, an application can immediately respond to the error and take the appropriate corrective action.

Because the RXRDY bit is not updated for data frames when sleep mode is enabled, the SCI can enable sleep mode and use a polling algorithm if desired. In this case, when RXRDY is set, software knows that a new address has been received. If the SCI is not being addressed, then the software should not change the value of the SLEEP bit and should continue to poll RXRDY.

22.6 LIN Communication Formats

The SCI/LIN module can be used in LIN mode or SCI mode. The enhancements for baud generation and additional receive/transmit buffers necessary for LIN mode operation are also part of the enhanced buffered SCI module. LIN mode is selected by enabling LIN MODE bit in SCIGCR1 register.

NOTE: The SCI/LIN is built around the SCI platform and uses a similar sampling scheme: 16 samples for each bit with majority vote on samples 8, 9, and 10.

The SCI/LIN control registers are located at the SCI/LIN base address. For a detailed description of each register, see Section 22.11.

22.6.1 LIN Standards

For compatibility with LIN2.0 standard the following additional features are implemented over LIN1.3:

i. Support for LIN 2.0 checksum
ii. Enhanced synchronizer FSM support for frame processing
iii. Enhanced handling of extended frames
iv. Enhanced baudrate generator
v. Update wakeup/go to sleep

The LIN module covers the CPU performance-consuming features, defined in the *LIN Specification Package* Revision 1.3 and 2.0 by hardware.

The Master Mode of LIN module is compatible with LIN 2.1 standard.

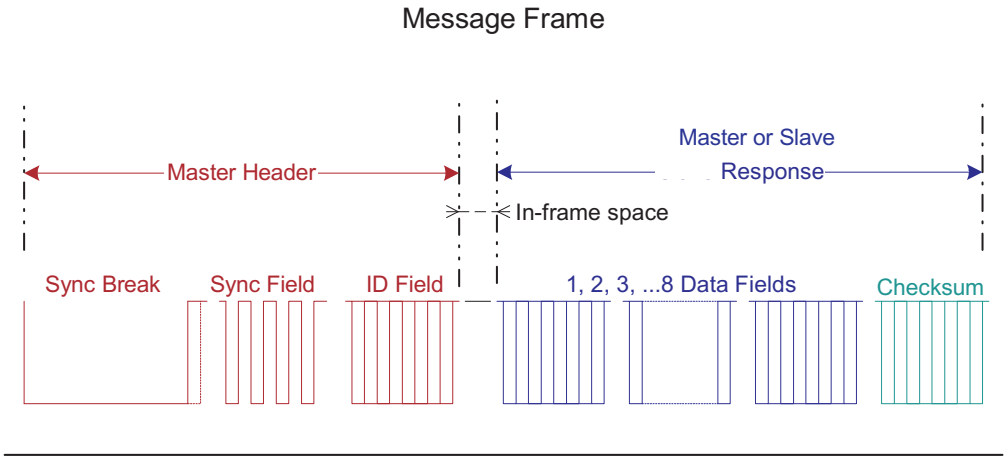
22.6.2 Message Frame

The LIN protocol defines a message frame format, illustrated in Figure 22-12. Each frame includes one master header, one response, one in-frame response space, and inter-byte spaces. In-frame-response and inter-byte spaces may be 0.

There is no arbitration in the definition of the LIN protocol; therefore, multiple slave nodes responding to a header might be detected as an error.

The LIN bus is a single channel wired-AND. The bus has a binary level: either dominant for a value of 0, or recessive for a value of 1.

Figure 22-12. LIN Protocol Message Frame Format: Master Header and Slave Response

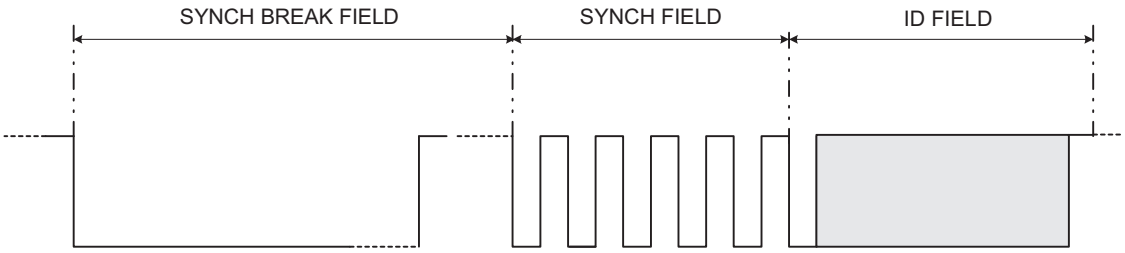


22.6.2.1 Message Header

The header of a message is initiated by a master (see Figure 22-13) and consists of a three field-sequence:

- The synch break field signaling the beginning of a message
- The synch field conveying bit rate information of the LIN bus
- The ID field denoting the content of a message

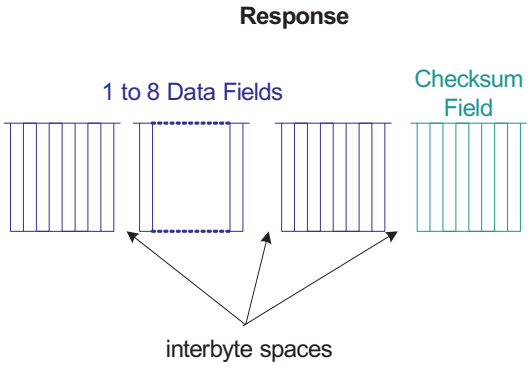
Figure 22-13. Header 3 Fields: Synch Break, Synch, and ID



22.6.2.2 Response

The format of the response is as illustrated in Figure 22-14. There are two types of fields in a response: data and checksum. The data field consists of exactly one data byte, one start bit, and one stop bit, for a total of 10 bits. The LSB is transmitted first. The checksum field consists of one checksum byte, one start bit and one stop bit. The checksum byte is the inverted modulo-256 sum over all data bytes in the data fields of the response.

Figure 22-14. Response Format of LIN Message Frame



The format of the response is a stream of N data fields and one checksum field. Typically N is from 1 to 8, with the exception of the extended command frames (Section 22.6.6). The length N of the response is indicated either with the optional length control bits of the ID Field (this is used in standards earlier than LIN 1.x); see Table 22-5, or by LENGTH value in SCIFORMAT[18:16] register; see Table 22-6. The SCI/LIN module supports response lengths from 1 to 8 bytes in compliance with LIN 2.0.

Table 22-5. Response Length Info Using IDBYTE Field Bits [5:4] for LIN Standards Earlier than 1.3

ID5	ID4	Number of Data bytes
0	0	2
0	1	2
1	0	4
1	1	8

Table 22-6. Response Length with SCIFORMAT[18:16] Programming

SCIFORMAT[18:16]	No. of Bytes
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

22.6.3 Synchronizer

The synchronizer has three major functions in the messaging between master and slave nodes. It generates the master header data stream, it synchronizes to the LIN bus for responding, and it locally detects timeouts. A bit rate is programmed using the prescalers in the BRS register to match the indicated LIN_speed value in the LIN description file.

The LIN synchronizer will perform the following functions: master header signal generation, slave detection and synchronization to message header with optional baud rate adjustment, response transmission timing and timeout control.

The LIN synchronizer is capable of detecting an incoming break and initializing communication at all times.

22.6.4 Baud Rate

The transmission baud rate of any node is configured by the CPU at the beginning; this defines the bit time T_{bit}. The bit time is derived from the fields P and M in the baud rate selection register (BRS). There is an additional 3-bit fractional divider value, field U in the BRS register, which further fine-tunes the data field baud rate.

The ranges for the prescaler values in the BRS register are:

P = 0, 1, 2, 3, . . . , 2^{24} - 1

M = 0, 1, 2, . . . , 15

U = 0, 1, 2, 3, 4, 5, 6, 7

The P, M, and U values in the BRS register are user programmable. The P and M dividers could be used for both SCI mode and LIN mode to select a baud rate. The U value is an additional 3-bit value determining that "**a TVCLK**" (with **a** = 0,1) is added to each T_{bit} as explained in Section 22.6.4.2. If the ADAPT bit is set and the LIN slave is in adaptive baud rate mode, then all these divider values are automatically obtained during header reception when the synchronization field is measured.

The LIN protocol defines baud rate boundaries as follows:

$1kHz \le F_{LINCLK} \le 20kHz$

All transmitted bits are shifted in and out at T_{bit} periods.

22.6.4.1 Fractional Divider

The M field in the BRS register modifies the integer prescaler P for fine tuning of the baud rate. The M value adds in increments of 1/16 of the P value.

The bit time, T_{bit} is expressed in terms of the VCLK period T_{VCLK} as follows:

For all P other than 0, and all M,

$$T_{bit} = 16\left(P + 1 + \frac{M}{16}\right)T_{VCLK} \quad (43)$$

For P= 0 : $T_{bit} = 32T_{VCLK}$

Therefore, the LINCLK frequency is given by:

$$F_{LINCLK} = \frac{F_{VCLK}}{16(P + 1 + \frac{M}{16})} \quad \text{For all P other than zero}$$

$$F_{LINCLK} = \frac{F_{VCLK}}{32} \quad \text{For } P = 0 \quad (44)$$

22.6.4.2 Superfractional Divider

The superfractional divider scheme applies to the following modes:

- LIN master mode (synch field + identifier field + response field + checksum field)
- LIN slave mode (response field + checksum field)

22.6.4.3 Superfractional Divider In LIN Mode

Building on the 4-bit fractional divider M (BRS[27:24], the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 22-7. The sync field (0x55), the identifier field and the response field can all be seen as 8-bit data bytes flanked by a start bit and a stop bit. The bits with a 1 in the table will have an additional VCLK period added to their T_{bit}.

Table 22-7. Superfractional Bit Modulation for LIN Master Mode and Slave Mode(1)

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0h	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0
2h	1	0	0	0	1	0	0	0	1	0
3h	1	0	1	0	1	0	0	0	1	0
4h	1	0	1	0	1	0	1	0	1	0
5h	1	1	1	0	1	0	1	0	1	1
6h	1	1	1	0	1	1	1	0	1	1
7h	1	1	1	1	1	1	1	0	1	1

(1)

1. In LIN master mode bit modulation applies to synch field + identifier field + response field
2. In LIN slave mode bit modulation applies to identifier field + response field

The baud rate will vary over a LIN data field to average according to the BRS[30:28] value by a *d* fraction of the peripheral internal clock: 0<d<1.

The instantaneous bit time is expressed in terms of T_{VCLK} *as follows:*

For all P other than 0, and all M and d (0 or 1),

$$T^{i}bit = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK} \quad (45)$$

For P = 0 $T_{bit} = 32T_{VCLK}$

The averaged bit time is expressed in terms of T_{VCLK} as follows.

For all P other than 0, and all M and d (0<d<1),

$$T^{a}bit = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK} \quad (46)$$

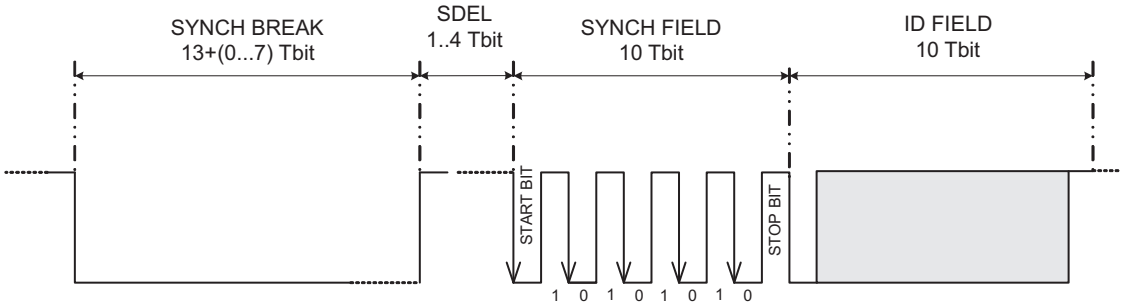
For P = 0 $T_{bit} = 32T_{VCLK}$

With the superfractional divider, a LIN baud rate of 20 kbps is achievable with an internal clock VCLK of 726 kHz. Furthermore, a rate of 400 kbps is achievable with an VCLK of 14.6 MHz.

22.6.5 Header Generation

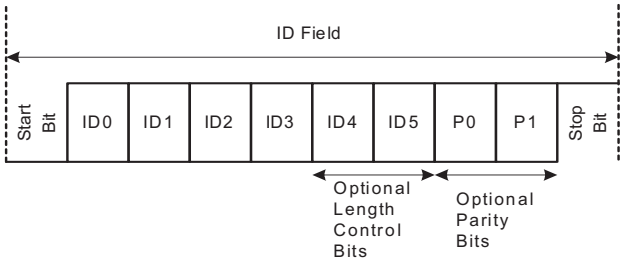
Automatic generation of the LIN protocol header data stream is supported without CPU interaction. The CPU will trigger a message header generation and the LIN state machine will handle the generation itself. A master node initiates header generation on CPU writes to the IDBYTE in the LINID register. The header is always sent by the master to initiate a LIN communication and consists of three fields: break field, synchronization field, and identification field, as seen in Figure 22-15.

Figure 22-15. Message Header in Terms of T_{bit}



- The break field consists of two components:
  - The synchronization break (SYNCH BREAK) consists of a minimum of 13 (dominant) low bits to a maximum of 20 dominant bits. The synch break length may be extended from the minimum with the 3-bit SBREAK value in the LINCOMP register.
  - The synchronization break delimiter (SDEL) consists of a minimum of 1 (recessive) high bit to a maximum of 4 recessive bits. The delimiter marks the end of the synchronization break field. The synch break delimiter length depends on the 2-bit SDEL value in the LINCOMP register.
- The synchronization field (SYNCH FIELD) consists of one start bit, byte 0x55, and a stop bit. It is used to convey T_{bit} information and resynchronize LIN bus nodes.
- The identifier field's ID byte may use six bits as an identifier, with optional length control (see *Note:Optional Control Length Bits*), and two optional bits as parity of the identifier. The identifier parity is used and checked if the PARITY ENA bit is set. If length control bits are not used, then there can be a total of 64 identifiers plus parity. If neither length control or parity are used there can be up to 256 identifiers. See Figure 22-16 for an illustration of the ID field.

Figure 22-16. ID Field



NOTE: Optional Control Length Bits

The control length bits only apply to LIN standards prior to LIN 1.3. IDBYTE field conveys response length information if compliant to standards earlier than LIN1.3. The SCIFORMAT register stores the length of the response for later versions of the LIN protocol.

NOTE: If the BLIN module, configured as Slave in multi-buffer mode, is in the process of transmitting data while a new header comes in, the module might end up in responding with the data from the previous interrupted response (not the data corresponding to the new ID). To avoid this scenario the following procedure could be used:

1. Check for the Bit Error (BE) during the response transmission. If the BE flag is set, this indicates that a collision has happened on the LIN bus (here because of the new Synch Break).
2. In the Bit Error ISR, configure the TD0 and TD1 registers with the next set of data to be transmitted on a TX Match for the incoming ID. Before writing to TD0/TD1 make sure that there was not already an update because of a Bit Error; otherwise TD0/TD1 might be written twice for one ID.
3. Once the complete ID is received, based on the match, the newly configured data will be transmitted by the node.

22.6.5.1 Event Triggered Frame Handling Proposal

The LIN 2.0 protocol uses event-triggered frames that may occasionally cause collisions. Event-triggered frames have to be handled in software.

If no slave answers to an event triggered frame header, the master node will set the NRE flag, and a NRE interrupt will occur if enabled. If a collision occurs, a frame error and checksum error may arise before the NRE error. Those errors are flagged and the appropriate interrupts will occur, if enabled.

Frame errors and checksum errors depend on the behavior and synchronization of the responding slaves. If the slaves are totally synchronized and stop transmission once the collision occurred, it is possible that only the NRE error is flagged despite the occurrence of a collision. To detect if there has been a reception of one byte before the NRE error is flagged, the BUS BUSY flag can be used as an indicator.

The bus busy flag is set on the reception of the first bit of the header and remains set until the header reception is complete, and again is set on the reception of the first bit of the response. In the case of a collision the flag is cleared in the same cycle as the NRE flag is set.

Software could implement the following sequence:

- Once the reception of the header is done (poll for RXID flag), wait for the bus busy flag to get set or NRE flag to get set.
- If bus busy flag is not set before NRE flag, then it is a true no response case (no data has been transmitted onto the bus).
- If bus busy flag gets set, then wait for NRE flag to get set or for successful reception. If NRE flag is set, then in this case a collision has occurred on the bus.

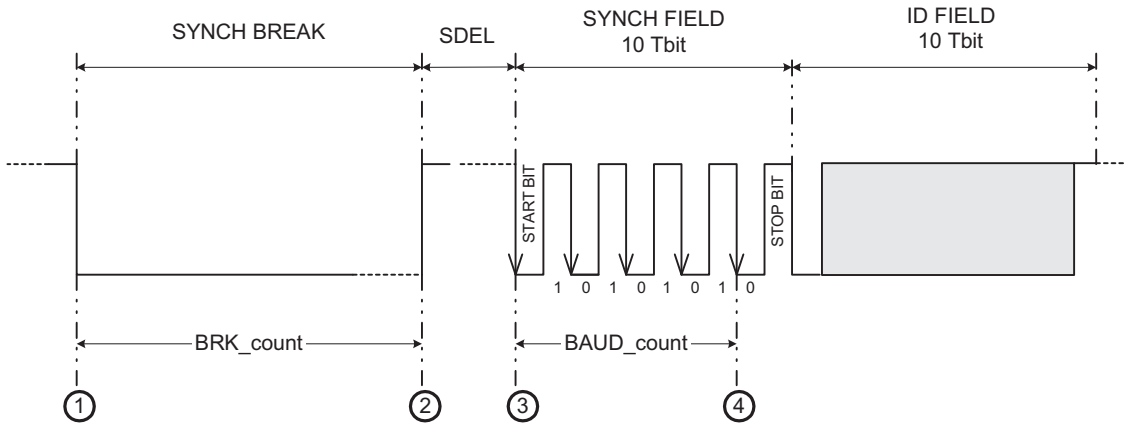
Even in the case of a collision, the received (corrupted) data is accessible in the RX buffers; registers LINRD0 and LINRD1.

22.6.5.2 Header Reception and Adaptive Baud Rate

A slave node baud rate can optionally be adjusted to the detected bit rate as an option to the LIN module. The adaptive baud rate option is enabled by setting the ADAPT bit. During header reception, a slave measures the baud rate during detection of the synch field. If ADAPT bit is set, then the measured baud rate is compared to the slave node's programmed baud rate and adjusted to the LIN bus baud rate if necessary.

The LIN synchronizer determines two measurements: BRK_count and BAUD_count (Figure 22-17). These values are always calculated during the Header reception for synch field validation (Figure 22-18).

Figure 22-17. Measurements for Synchronization



By measuring the values BRK_count and BAUD_count, a valid synch break sequence can be detected as described in Figure 22-18. The four numbered events in Figure 22-17 signal the start/stop of the synchronizer counter. The synchronizer counter uses VCLK as the time base.

The synchronizer counter is used to measure the synch break relative to the detecting node T_{bit}. For a slave node receiving the synch break, a threshold of 11 T_{bit} is used as required by the LIN protocol. For detection of the dominant data stream of the synch break, the synchronizer counter is started on a falling edge and stopped on a rising edge of the LINRX. On detection of the synch break delimiter, the synchronizer counter value is saved and then reset.

On detection of five consecutive falling edges, the BAUD_count is measured. Bit timing calculation and consistency to required accuracy is implemented following the recommendations of LIN revision 2.0. A slave node can calculate a single T_{bit} time by division of BAUD_count by 8. In addition, for consistency between the detected edges the following is evaluated:

BAUD_count + BAUD_count » 2 + BAUD_count » 3 ≤ BRK_count

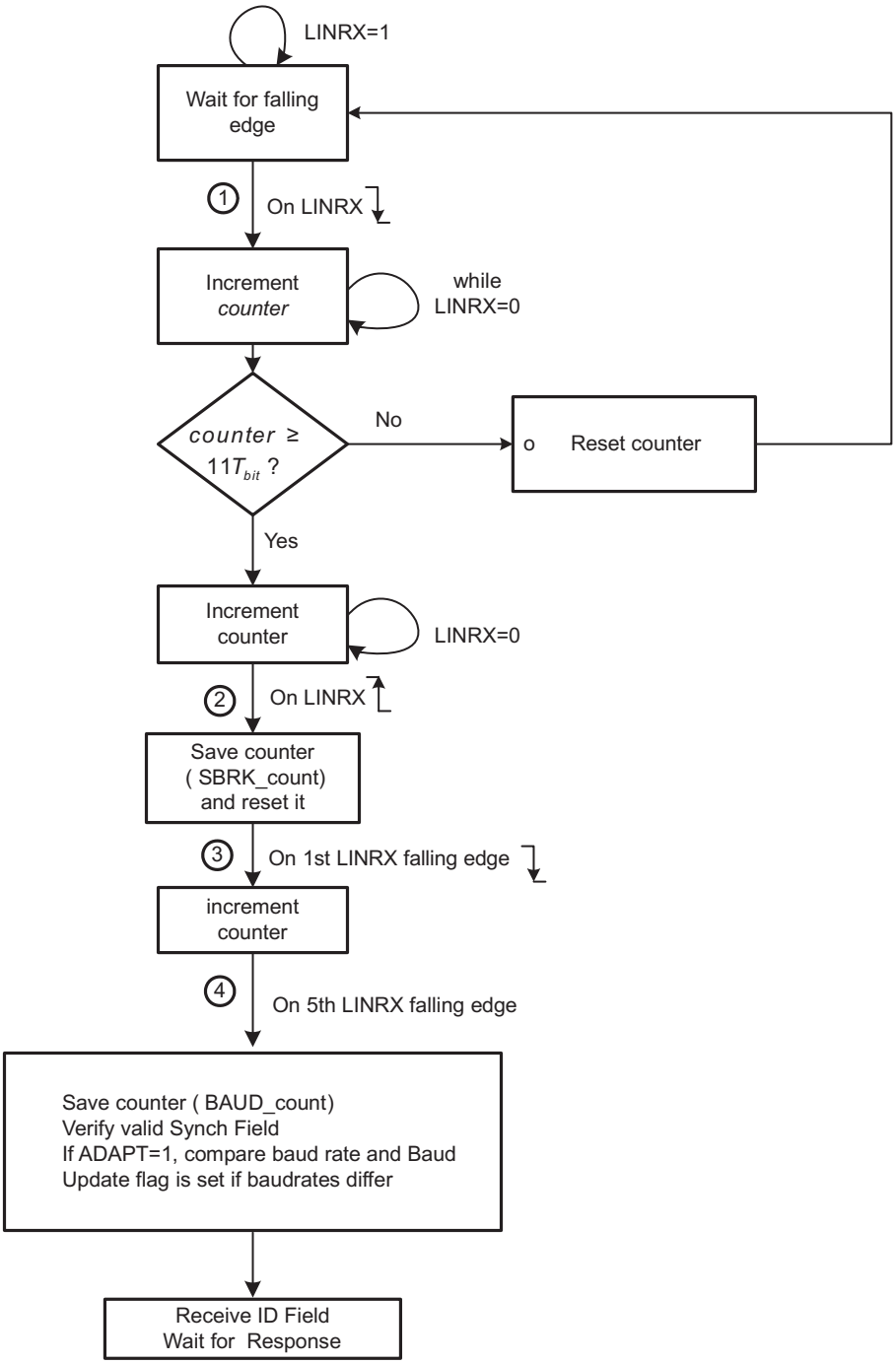
The BAUD_count value is shifted 3 times to the right and rounded using the first insignificant bit to obtain a T_{bit} unit. If the ADAPT bit is set, then the detected baud rate is compared to the programmed baud rate.

During the header reception processing as illustrated in Figure 22-18, if the measured BRK_count value is less than 11 T_{bit}, the synch break is not valid according to the protocol for a fixed rate. If the ADAPT bit is set, then the MBRS register is used for measuring BRK_count and BAUD_count values and automatically adjusts to any allowed LIN bus rate (refer to *LIN Specification Package 2.0*).

NOTE: In adaptive mode the MBRS divider should be set to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise a 0x00 data byte could mistakenly be detected as a synch break.

The break-threshold relative to the slave node is 11 T_{bit}. The break is 13 T_{bit} as specified in LIN version 1.3.

Figure 22-18. Synchronization Validation Process and Baud Rate Adjustment



If the synch field is not detected within the given tolerances, the inconsistent-synch-field-error (ISFE) flag will be set. An ISFE interrupt will be generated, if enabled by its respective bit in the SCISETINT register. The ID byte should be received after the synch field validation was successful. Any time a valid break (larger than 11 T_{bit}) is detected, the receiver's state machine should reset to reception of this new frame. This reset condition is only valid during response state, not if an additional synch break occurs during header reception.

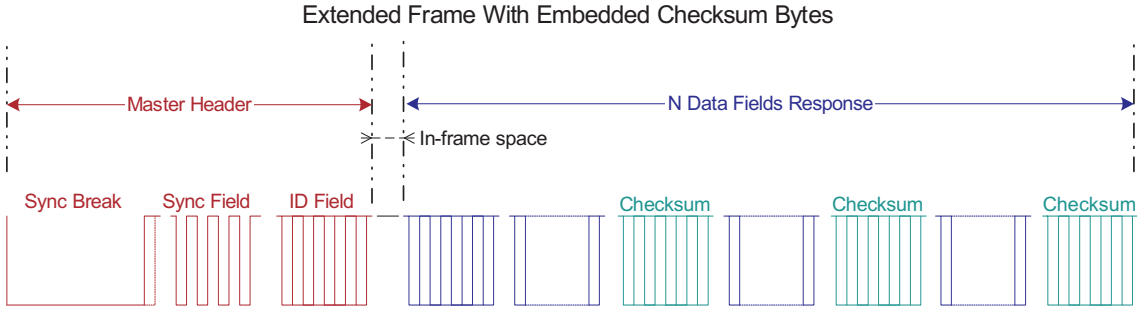
NOTE: When an inconsistent synch field (ISFE) error occurs, suggested action for the application is to Reset the SWnRST bit and Set the SWnRST bit to make sure that the internal state machines are back to their normal states

22.6.6 Extended Frames Handling

The LIN protocol 2.0 and prior includes two extended frames with identifiers 62 (user defined) and 63 (reserved extended). The response data length of the user-defined frame (ID 62, or 0x3E) is unlimited. The length for this identifier will be set at network configuration time to be shared with the LIN bus nodes.

Extended frame communication is triggered on reception of a header with identifier 0x3E; see Figure 22-19. Once the extended frame communication is triggered, unlike normal frames, this communication needs to be stopped before issuing another header. To stop the extended frame communication the STOP EXT FRAME bit must be set.

Figure 22-19. Optional Embedded Checksum in Response for Extended Frames



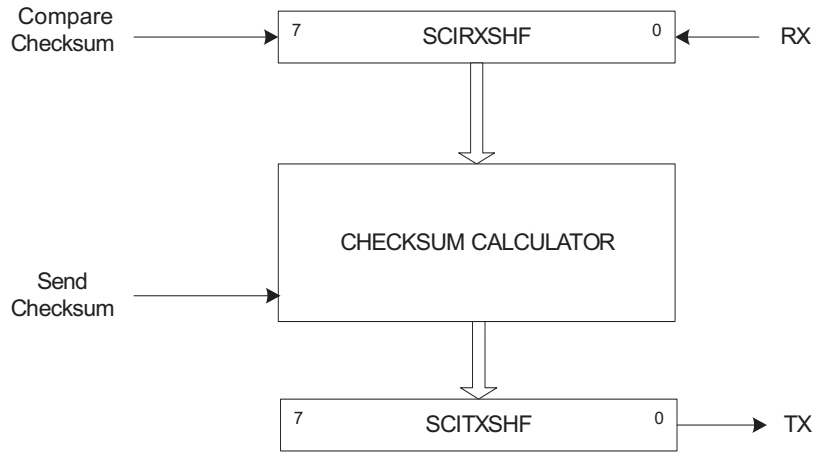
An ID interrupt will be generated (if enabled and there is a match) on reception of ID 62 (0x3E). This interrupt allows the CPU using a software counter to keep track of the bytes that are being sent out and decides when to calculate and insert a checksum byte (recommended at periodic rates). To handle this procedure, SC bit is used. A write to the send checksum bit SC will initiate an automatic send of the checksum byte. The last data field should always be a checksum in compliance with the LIN protocol.

The periodicity of the checksum insertion, defined at network configuration time, is used by the receiving node to evaluate the checksum of the ongoing message, and has the benefit of enhanced reliability.

For the sending node, the checksum is automatically embedded each time the send checksum bit SC is set. For the receiving node, the checksum is compared each time the compare checksum bit CC is set; see Figure 22-20.

NOTE: The LIN 2.0 enhanced checksum does not apply to the reserved identifiers. The reserved identifiers always use the classic checksum.

Figure 22-20. Checksum Compare and Send for Extended Frames



22.6.7 Timeout Control

Any LIN node listening to the bus and expecting a response initiated from a master node could flag a no-response error timeout event. The LIN protocol defines four types of timeout events, which are all handled by the hardware of the LIN module. The four LIN protocol events are:

- No-response timeout error
- Bus idle detection
- Timeout after wakeup signal
- Timeout after three wakeup signals

22.6.7.1 No-Response Error (NRE)

The no-response error will occur when any node expecting a response waits for T_{FRAME_MAX} time and the message frame is not fully completed within the maximum length allowed, T_{FRAME_MAX}. After this time a no-response error (NRE) is flagged in the NRE bit of the SCIFLR register. An interrupt is triggered if enabled.

As specified in the LIN 1.3 standard, the minimum time to transmit a frame is:

$T_{FRAME\_MIN} = T_{HEADER\_MIN} + T_{DATA\_FIELD} + T_{CHECKSUM\_FIELD}$

$= 44 + 10N$

where N = number of data fields.

And the maximum time frame is given by:

$T_{FRAME\_MAX} = T_{FRAME\_MIN} * 1.4$

$= (44 + 10N) * 1.4$

The timeout value T_{FRAME_MAX} is derived from the *N* number of data fields value. The *N* value is either embedded in the header's ID field for messages or is part of the description file. In the latter case, the 3-bit CHAR value in SCIFORMAT register, will indicate the value for *N*.

NOTE: The length coding of the ID field does not apply to two extended frame identifiers, ID fields of 0x3E (62) and 0x3F (63). In these cases, the ID field can be followed by an arbitrary number of data byte fields. Also, the LIN 2.0 protocol specification mentions that ID field 0x3F (63) cannot be used. For these two cases, the NRE will not be handled by the LIN controller hardware.

Table 22-8. Timeout Values in T_{bit} Units

N	T_{DATA_FIELD}	T_{FRAME_MIN}	T_{FRAME_MAX}
1	10	54	76
2	20	64	90
3	30	74	104
4	40	84	118
5	50	94	132
6	60	104	146
7	70	114	160
8	80	124	174

22.6.7.2 Bus Idle Detection

The second type of timeout can occur when a node detects an inactive LIN bus: no transitions between recessive and dominant values are detected on the bus. This happens after a minimum of 4 s (this is 80,000 F_{LINCLK} cycles with the fastest bus rate of 20 kbps). If a node detects no activity in the bus as the TIMEOUT bit is set, then it can be assumed that the LIN bus is in sleep mode. Application software can use the Timeout flag to determine when the LIN bus is inactive and put the LIN into sleep mode by writing the POWERDOWN bit.

NOTE: After the timeout was flagged, a SW nRESET should be asserted before entering Low-Power Mode. This is required to reset the receiver in case that an incomplete frame was on the bus before the idle period.

22.6.7.3 Timeout after Wakeup Signal and Timeout after Three Wakeup Signals

The third and fourth types of timeout are related to the wakeup signal. A node initiating a wakeup should expect a header from the master within a defined amount of time: timeout after wakeup signal. See Section 22.9.3 for more details.

22.6.8 TXRX Error Detector (TED)

The following sources of error are detected by the TXRX error detector logic (TED). The TED logic consists of a bit monitor, an ID parity checker, and a checksum error. The following errors are detected:

- Bit errors (BE)
- Physical bus errors (PBE)
- Identifier parity errors (PE)
- Checksum errors (CE)

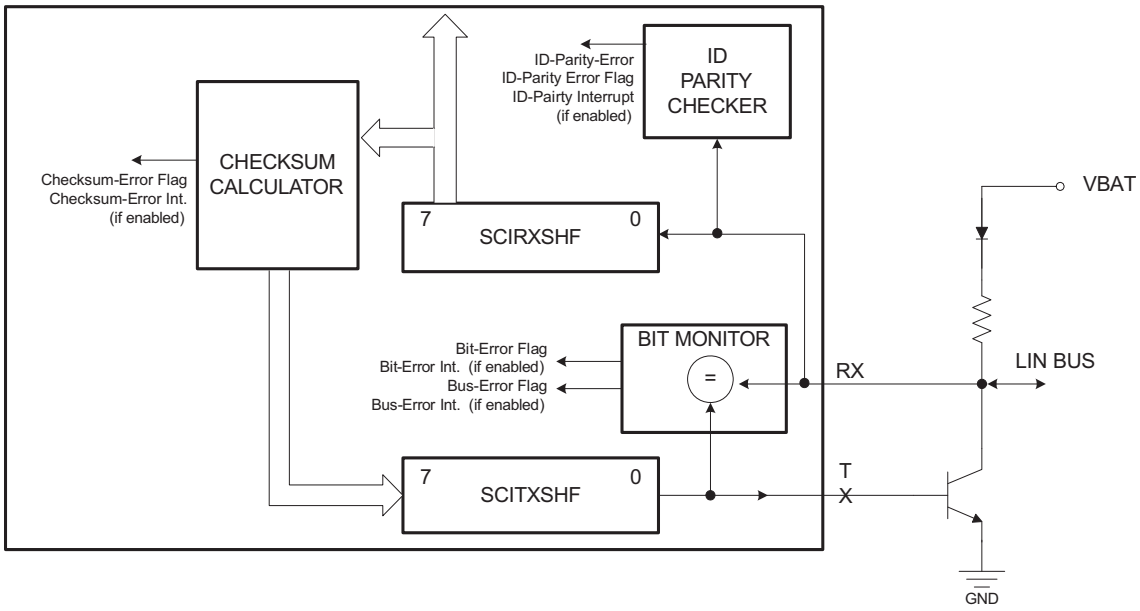
All of these errors (BE, PBE, PE, CE) are flagged. An interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame.

22.6.8.1 Bit Errors

A bit error (BE) is detected at the bit time when the bit value that is monitored is different from the bit value that is sent. A bit error is indicated by the BE flag in SCIFLR. After signaling a BE, the transmission is aborted no later than the next byte. The bit monitor ensures that the transmitted bit in LINTX is the correct value on the LIN bus by reading back on the LINRX pin as shown in Figure 22-21.

NOTE: If BE Occurs due to New Header reception during a Slave Response, NRE/TIMEOUT flag will not be set for the new Frame.

Figure 22-21. TXRX Error Detector



22.6.8.2 Physical Bus Errors

A Physical Bus Error (PBE) has to be detected by a master if no valid message can be generated on the bus (Bus shorted to GND or VBAT). The bit monitor detects a PBE during the header transmission, if no Synch Break can be generated (for example, because of a bus shortage to VBAT) or if no Synch break Delimiter can be generated (for example, because of a bus shortage to GND). Once the Synch Break Delimiter was validated, all other deviations between the monitored and the sent bit value are flagged as Bit Errors (BE) for this frame.

22.6.8.3 ID Parity Errors

If parity is enabled, an ID parity error (PE) is detected if any of the two parity bits of the sent ID byte are not equal to the calculated parity on the receiver node. The two parity bits are generated using the following mixed parity algorithm.

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \ (even\ Parity)$$
$$P1 = ID1 \oplus ID3 \oplus ID4 \oplus ID5 \ (odd\ \ Parity) \quad (47)$$

If an ID-parity error is detected, the ID-parity error is flagged, and the received ID is not valid. See Section 22.6.9 for details.

22.6.8.4 Checksum Errors

A checksum error (CE) is detected and flagged at the receiving end if the calculated modulo-256 sum over all received data bytes (including the ID byte if it is the enhanced checksum type) plus the checksum byte does not result in 0xFF. The modulo-256 sum is calculated over each byte by adding with carry, where the carry bit of each addition is added to the LSB of its resulting sum.

For the transmitting node, the checksum byte sent at the end of a message is the inverted sum of all the data bytes (see Figure 22-22) for classic checksum implementation. The checksum byte is the inverted sum of the identifier byte and all the data bytes (see Figure 22-23) for the LIN 2.0 compliant enhanced checksum implementation. The classic checksum implementation should always be used for reserved identifiers 60 to 63; therefore, the CTYPE bit will be overridden in this case. For signal-carrying-frame identifiers (0 to 59) the type of checksum used depends on the CTYPE bit.

Figure 22-22. Classic Checksum Generation at Transmitting Node

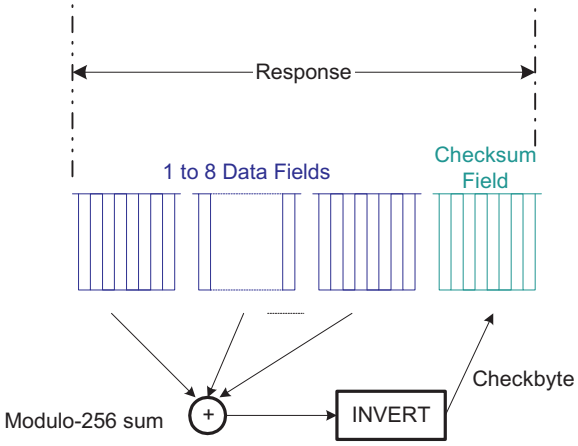
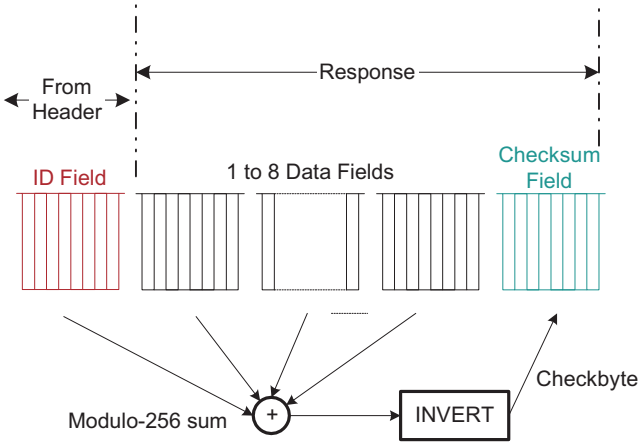


Figure 22-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node



22.6.9 Message Filtering and Validation

Message filtering uses the entire identifier to determine which nodes will participate in a response, either receiving or transmitting a response. Therefore, two acceptance masks are used, as shown in Figure 22-24. During header reception, all nodes filter the ID-Field (ID-Field is the part of the header explained in Figure 22-16) to determine whether they transmit a response or receive a response for the current message. There are two masks for message ID filtering: one to accept a response reception, the other to initiate a response transmission. See Figure 22-24. All nodes compare the received ID to the identifier stored in the ID-SlaveTask BYTE of the LINID register and use the RX ID MASK and the TX ID MASK fields in the LINMASK register to filter the bits of the identifier that should not be compared.

If there is an RX match with no parity error and the RXENA bit is set, there will be an ID RX flag and an interrupt will be triggered if enabled. If there is a TX match with no parity error and the TXENA bit is set, there will be an ID TX flag and an interrupt will be triggered if enabled in the SCISETINT register.

The masked bits become don't cares for the comparison. To build a mask for a set of identifiers, an XOR function could be used.

For example, to build a mask to accept IDs 0x26 and 0x25 using LINID[7:0] = 0x20; that is, compare 5 most significant bits (MSBs) and filter 3 least significant bits (LSBs), the acceptance mask could be:

$$(0x26 + 0x25) \oplus 0x20 = 0x07 \quad (48)$$

A mask of all zeros will compare all bits of the received identifier in the shift register with the ID-BYTE in LINID[7:0]. If HGEN CTRL is set to 1, a mask of 0xFF will always cause a match. A mask of all 1s will filter all bits of the received identifier, and thus there will be an ID match regardless of the content of the ID-SlaveTask BYTE field in the LINID register.

NOTE: When the HGEN CTRL bit = 0, the LIN nodes compare the received ID to the ID-BYTE field in the LINID register, and use the RX ID MASK and the TX ID MASK in the LINMASK register to filter the bits of the identifier that should not be compared.

If there is an RX match with no parity error and the RXENA bit is set, there will be an ID RX flag and an interrupt will be triggered if enabled. A mask of all 0s will compare all bits of the received identifier in the shift register with the ID-BYTE field in LINID[7:0]. A mask of all 1s will filter all bits of the received identifier and there will be no match.

During header reception, the received identifier is copied to the Received ID field LINID[23:16]. If there is no parity error and there is either a TX match or an RX match, then the corresponding TX or RX ID flag is set. If the ID interrupt is enabled, then an ID interrupt is generated.

After the ID interrupt is generated, the CPU may read the Received ID field LINID[23:16] and determine what response to load into the transmit buffers.

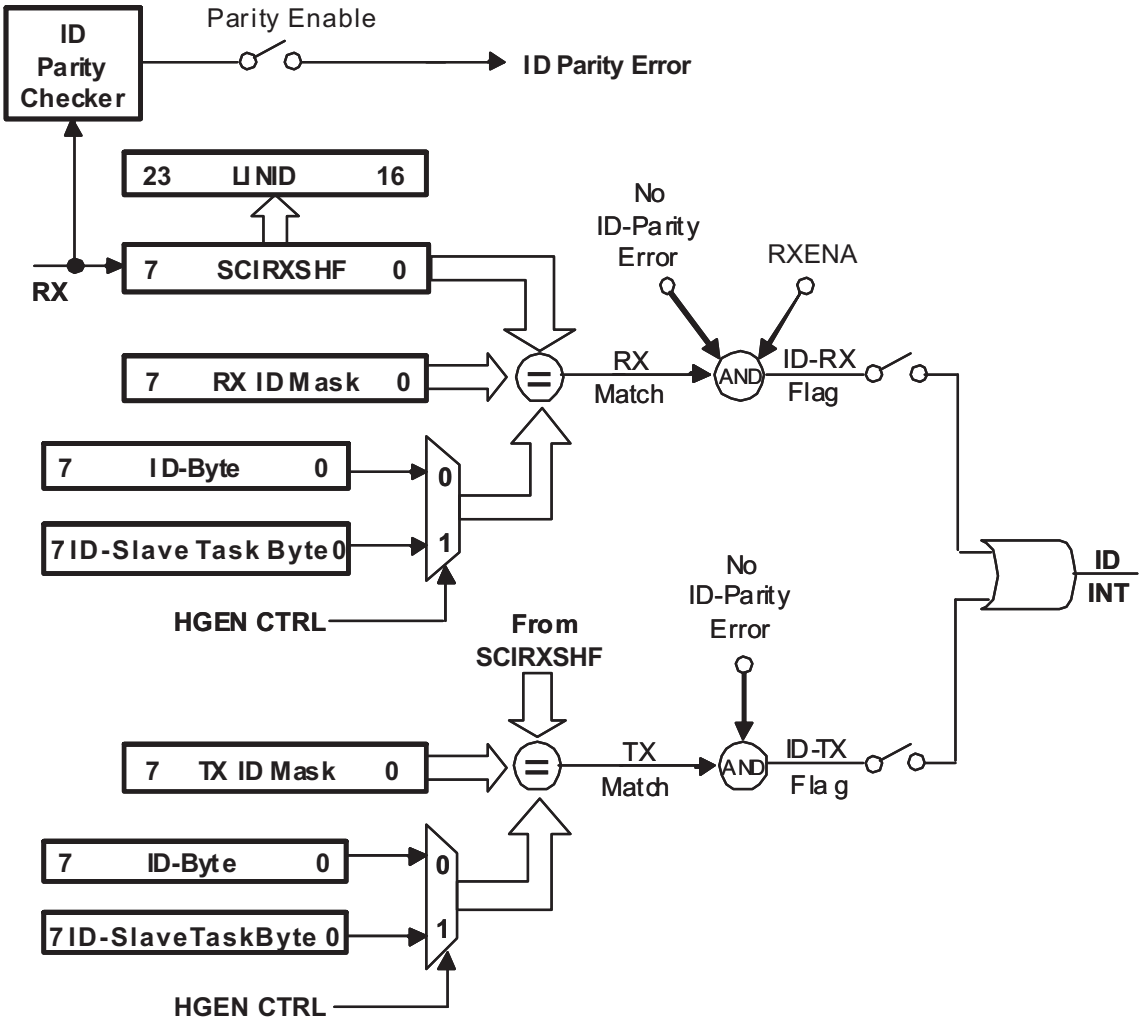
NOTE: When byte 0 is written to TD0 (LINTD0[31:24]), the response transmission is automatically generated.

In multi-buffer mode, the TXRDY flag will be set when all the response data bytes and checksum byte are copied to the shift register SCITXSHF. In non multi-buffer mode, the TXRDY flag is set each time a byte is copied to the SCITXSHF register, and also for the last byte of the frame after the checksum byte is copied to the SCITXSHF register.

In multi-buffer mode, the TXEMPTY flag is set when both the transmit buffer(s) TDy and the SCITXSHF shift register are emptied and the checksum has been sent. In non multi-buffer mode, TXEMPTY is set each time TD0 and SCITXSHF are emptied, except for the last byte of the frame where the checksum byte must also be transmitted.

If parity is enabled, all slave receiving nodes will validate the identifier using all eight bits of the received ID byte. The SCI/LIN will flag a corrupted identifier if an ID-parity error is detected.

Figure 22-24. ID Reception, Filtering and Validation



22.6.10 Receive Buffers

To reduce CPU load when receiving a LIN N-byte (with N = 1–8) response in interrupt mode , the SCI/LIN module has eight receive buffers. These buffers can store an entire LIN response in the RDy receive buffers.

The checksum byte following the data bytes is validated by the internal checksum calculator. The checksum error (CE) flag indicates a checksum error and a CE interrupt will be generated if enabled in the SCISETINT register.

The multi-buffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers if multi-buffer mode is enabled, or to RD0 if multi-buffer mode is disabled. The 3-bit compare register contains the number of data bytes expected to be received. In cases where the ID BYTE field does not convey message length (see *Note:Optional Control Length Bits* in Section 22.6.5), the LENGTH value, indicates the expected length and is used to load the 3-bit compare register. Whether the length control field or the LENGTH value is used is selectable with the COMM MODE bit.

Figure 22-8 illustrates the receive buffers.

A receive interrupt and a receive ready RXRDY flag set could occur after receiving a response if there are no response receive errors for the frame (such as, there is no checksum error, frame error, and overrun error). The checksum byte will be compared before acknowledging a reception.

NOTE: In multi-buffer mode following are the scenarios associated with clearing the "RXRDY" flag bit:

1. The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.
2. For LENGTH less than or equal to 4, Read to RD0 register will clear the “RXRDY” flag.
3. For LENGTH greater than 4, Read to RD1 register will clear the “RXRDY” flag.

22.6.11 Transmit Buffers

To reduce the CPU load when transmitting a LIN N-byte (with N = 1–8) response in interrupt mode, the SCI/LIN module has eight transmit buffers, TD0–TD7 in LINTD0 and LINTD1. With these transmit buffers, an entire LIN response field can be pre-loaded in the TXy transmit buffers.

The multi-buffer 3-bit counter counts the data bytes transferred from the TDy transmit buffers register if multi-buffer mode is enabled, or from TD0 to SCITXSHF if multi-buffer mode is disabled. The 3-bit compare register contains the number of data bytes expected to be transmitted. If the ID field is not used to convey message length (see *Note: Optional Control Length Bits* in Section 22.6.5), the LENGTH value indicates the expected length and is used instead to load the 3-bit compare register. Whether the length control field or the LENGTH value is used is selectable with the COMM MODE bit.

A transmit interrupt (TX interrupt), and a transmit ready flag (TXRDY flag) could occur after transmitting a response.

Figure 22-9 illustrates the transmit buffers.

The checksum byte will be automatically generated by the checksum calculator and sent after the data-fields transmission is finished. The multi-buffer 3-bit counter counts the data bytes transferred from the TDy buffers into the SCITXSHF register.

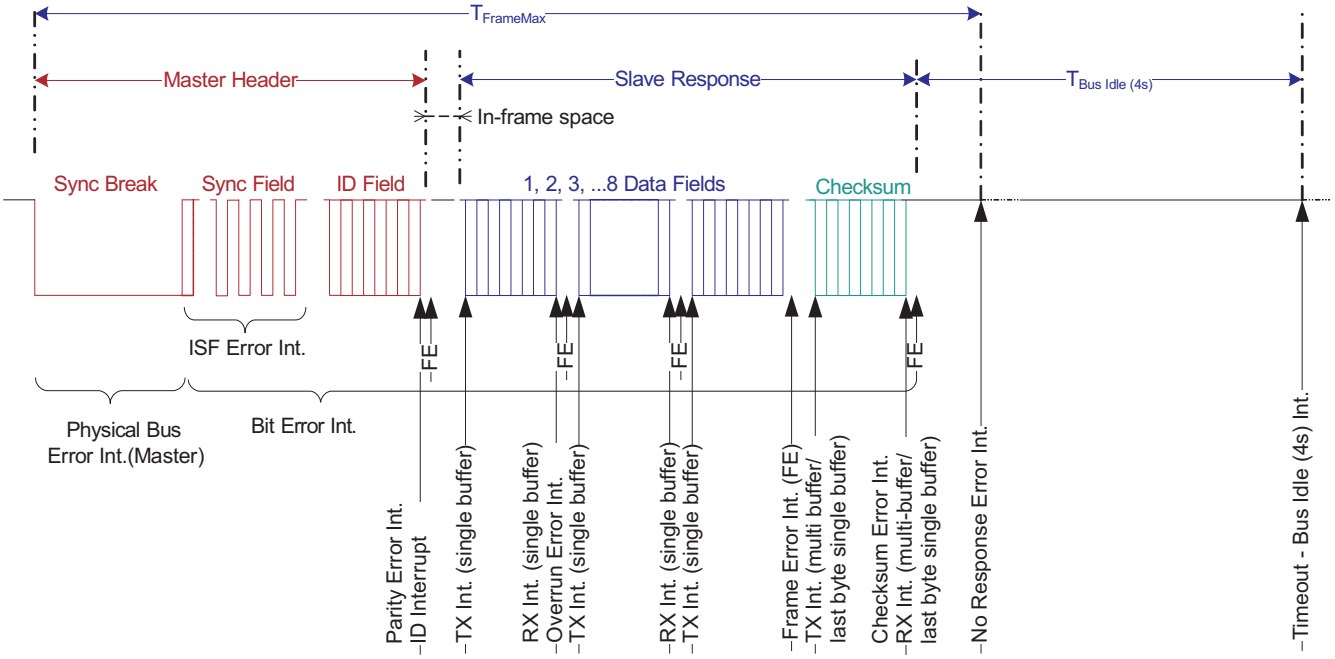
NOTE: The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLRINT register or by disabling the transmitter via the TXENA bit.

22.7 LIN Interrupts

LIN and SCI mode have a common Interrupt block as explained in Section 22.3. There are 16 interrupt sources in the SCI/LIN module, with 8 of them being LIN mode only, as seen in Table 22-4.

A LIN message frame indicating the timing and sequence of the LIN interrupts that could occur is shown in Figure 22-25.

Figure 22-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence



22.8 LIN Configurations

The following list details the configuration steps that software should perform prior to the transmission or reception of data in LIN mode. As long as the SWnRST bit in the SCIGCR1 register is cleared to 0 the entire time that the LIN is being configured, the order in which the registers are programmed is not important.

- Enable LIN by setting the RESET bit in SCIGCR0 to 1.
- Clear the SWnRST bit to 0 before LIN is configured.
- Select the desired frame format by programming the SCIGCR1 register.
- Set both the RX FUNC and TX FUNC bits in SCIPIO0 to 1 to configure the LINRX and LINTX pins for LIN functionality.
- Set the LIN MODE bit in SCIGCR1 to 1 to enable LIN mode.
- Select Master or Slave mode by programming the CLOCK bit in SCIGCR1.
- Set the MBUF MODE bit in SCIGCR1 to 1 to select multi-buffer mode.
- Select the baud rate to be used for communication by programming the BRS register.
- Set the maximum baud rate to be used for communication by programming the MBRS register.
- Set the CONT bit in SCIGCR1 to 1 to make LIN not halt for an emulation breakpoint until its current reception or transmission is complete (this bit is used only in an emulation environment).
- Set LOOP BACK bit in SCIGCR1 to 1 to connect the transmitter to the receiver internally (this feature is used to perform a self-test).
- Set the RXENA bit in SCIGCR1 to 1, if data is to be received.
- Set the TXENA bit in SCIGCR1 to 1, if data is to be transmitted.
- Select the RX ID MASK and the TX ID MASK fields in the LINMASK register.
- Set the SWnRST bit to 1 after LIN is configured.
- Perform receiving or transmitting data (see Section 22.8.1 or Section 22.8.2).

22.8.1 Receiving Data

The LIN receiver is enabled to receive messages if both the RX FUNC bit and the RXENA bit are set to 1. If the RX FUNC bit is not set, the LINRX pin functions as a general-purpose I/O pin rather than as a LIN function pin.

The ID RX FLAG is set after a valid LIN ID is received with RX Match. An ID interrupt is generated, if enabled.

22.8.1.1 Receiving Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit is cleared to 0. In this mode, LIN sets the RXRDY bit when it transfers newly received data from SCIRXSHF to RD0. The SCI clears the RXRDY bit after the new data in RD0 has been read. Also, as data is transferred from SCIRXSHF to RD0, the LIN sets the FE, OE, or PE flags if any of these error conditions were detected in the received data. These error conditions are supported with configurable interrupt capability.

You can receive data by:

1. Polling Receive Ready Flag
2. Receive Interrupt

In polling method, software can poll for the RXRDY bit and read the data from RD0 byte of the LINRD0 register once the RXRDY bit is set high. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use the interrupt method. To use the interrupt method, the SET RX INT bit is set. An interrupt is generated the moment the RXRDY bit is set. If the checksum scheme is enabled by setting the Compare Checksum (CC) bit to 1, the checksum will be compared on the byte that is currently being received, which is expected to be the checksum byte. The CC bit will be cleared once the checksum is received. A CE will immediately be flagged if there is a checksum error.

22.8.1.2 Receiving Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit is set to 1. In this mode, LIN sets the RXRDY bit after receiving the programmed number of data in the receive buffer and the checksum field, the complete frame. The error condition detection logic is similar to the single-buffer mode, except that it monitors for the complete frame. Like single-buffer mode, you can use the polling or interrupt method to read the data. The received data has to be read from the LINRD0 and LINRD1 registers, based on the number of bytes. For a LENGTH less than or equal to 4, a read from the LINRD0 register clears the RXRDY flag. For a LENGTH greater than 4, a read from the LINRD1 register clears the RXRDY flag. If the checksum scheme is enabled by setting the Compare Checksum (CC) bit to 1 during the reception of the data, then the byte that is received after the reception of the programmed number of data bytes indicated by the LENGTH field is treated as a checksum byte. The CC bit will be cleared once the checksum is received and compared.

22.8.2 Transmitting Data

The LIN transmitter is enabled if both the TX FUNC bit and the TXENA bit are set to 1. If the TX FUNC bit is not set, the LINTX pin functions as a general-purpose I/O pin rather than as a LIN function pin. Any value written to the TD0 before the TXENA bit is set to 1 is not transmitted. Both of these control bits allow for the LIN transmitter to be held inactive independently of the receiver.

The ID TX flag is set after a valid LIN ID is received with TX Match. An ID interrupt is generated, if enabled.

22.8.2.1 Transmitting Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit is cleared to 0. In this mode, LIN waits for data to be written to TD0, transfers it to SCITXSHF, and transmits the data. The TXRDY and TX EMPTY bits indicate the status of the transmit buffers. That is, when the transmitter is ready for data to be written to TD0, the TXRDY bit is set. Additionally, if both TD0 and SCITXSHF are empty, then the TX EMPTY bit is also set.

You can transmit data by:

1. Polling Transmit Ready Flag
2. Transmit Interrupt

In polling method, software can poll for the TXRDY bit to go high before writing the data to the TD0. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use the interrupt method. To use the interrupt method, the SET TX INT bit is set. An interrupt is generated the moment the TXRDY bit is set. When the LIN has completed transmission of all pending frames, the SCITXSHF register and the TD0 are empty, the TXRDY bit is set, and an interrupt is generated, if enabled. Because all data has been transmitted, the interrupt should be halted. This can either be done by disabling the transmit interrupt (CLR TX INT) or by disabling the transmitter (clear TXENA bit). If the checksum scheme is enabled by setting the Send Checksum (SC) bit to 1, the checksum byte will be sent after the current byte transmission. The SC bit will be cleared after the checksum byte has been transmitted.

NOTE: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

22.8.2.2 Transmitting Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit is set to 1. Like single-buffer mode, you can use the polling or interrupt method to write the data to be transmitted. The transmitted data has to be written to the LINTD0 and LINTD1 registers, based on the number of bytes. LIN waits for data to be written to Byte 0 (TD0) of the LINTD0 register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically. If the checksum scheme is enabled by setting the Send Checksum (SC) bit to 1, the checksum will be sent after transmission of the last byte of the programmed number of data bytes, indicated by the LENGTH field. The SC bit will be cleared after the checksum byte has been transmitted.

22.9 Low-Power Mode

The SCI/LIN module can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SCI/LIN module. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive. If global low-power mode is requested while the receiver is receiving data, then the SCI/BLIN completes the current reception and then enters the low-power mode, that is, module enters low-power mode only when Busy bit (SCIFLR.3) is cleared.

The BLIN module may enter low-power mode either when there was no activity on the LINRX pin for more than 4s (this can be either a constant recessive or dominant level) or when a Sleep Command frame was received. Once the Timeout flag (SCIFLR.4) was set or once a Sleep Command was received, the POWERDOWN bit (SCIGCR2.0) must be set by the application software to make the module enter local low-power mode. A wakeup signal will terminate the sleep mode of the LIN bus.

NOTE: Enabling Local Low-Power Mode During Receive and Transmit

If the wakeup interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI/LIN immediately generates a wake-up interrupt to clear the powerdown bit. Thus, the SCI/LIN is prevented from entering low-power mode and completes the current reception. Otherwise, if the wakeup interrupt is disabled, the SCI/LIN completes the current reception and then enters the low-power mode.

22.9.1 Entering Sleep Mode

In LIN protocol, a sleep command is used to broadcast the sleep mode to all nodes. The sleep command consists of a diagnostic master request frame with identifier 0x3C (60), with the first data field as 0x00. There should be no activity in the bus once all nodes receive the sleep command: the bus is in sleep mode.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and registers. Clearing the POWERDOWN bit causes SCI/LIN to exit from local low-power mode. All the registers are accessible during local power-down mode. If a register is accessed in low-power mode, this access results in enabling the clock to the module for that particular access alone.

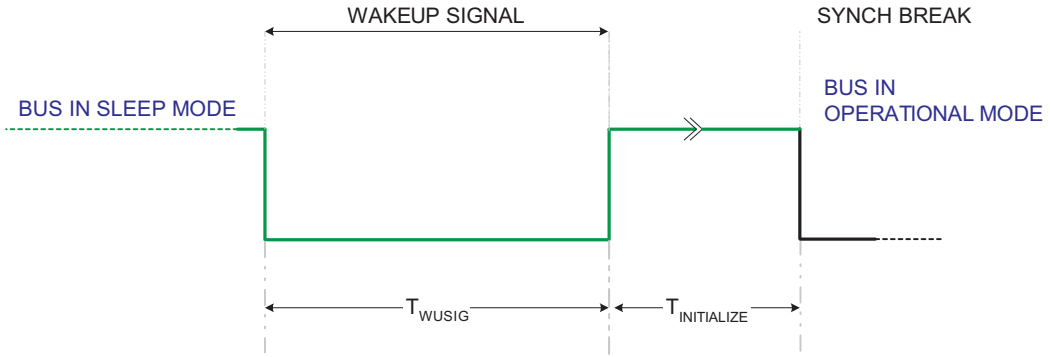
22.9.2 Wakeup

The wakeup interrupt is used to allow the SCI/LIN module to automatically exit low-power mode. A SCI/LIN wakeup is triggered when a low level is detected on the receive RX pin, and this clears the POWERDOWN bit.

NOTE: If the wakeup interrupt is disabled then the SCI/LIN enters low-power mode whenever it is requested to do so, but a low level on the receive RX pin does NOT cause the SCI/LIN to exit low-power mode.

In LIN mode, any node can terminate sleep mode by sending a wakeup signal; see Figure 22-26. A slave node that detects the bus in sleep mode, and with a wakeup request pending, will send a wakeup signal. The wakeup signal is a dominant value on the LIN bus for T_{WUSIG}; this is at least 5 T_{bits} for the LIN bus baud rates. The wakeup signal is generated by sending an 0xF0 byte containing 5 dominant T_{bits} and 5 recessive T_{bits}.

Figure 22-26. Wakeup Signal Generation



$$0.25ms \le T_{WUSIG} \le 5ms \quad (49)$$

Assuming a perfect bus with no noise or loading effects, a write of 0xF0 to TD0 will load the transmitter to meet the wakeup signal timing requirement for T_{WUSIG}. Then, setting the GENWU bit will transmit the preloaded value in TD0 for a wakeup signal transmission.

NOTE: The GENWU bit can be set/reset only when SWnRST is set to'1' and the node is in power down mode. The bit will be cleared on a valid synch break detection. A master sending a wakeup request, will exit power down mode upon reception of the wakeup pulse. The bit will be cleared on a SWnRST. This can be used to stop a master from sending further wakeup requests.

The TI TPIC1021 LIN transceiver, upon receiving a wakeup signal, will translate it to the microcontroller for wakeup with a dominant level on the RX pin, or a signal to the voltage regulator. While the POWERDOWN bit is set, if the LIN module detects a recessive-to-dominant edge (falling edge) on the RX pin, it will generate a wakeup interrupt if enabled in the SCISETINT register.

According to LIN protocol 2.0, the TI TPIC1021 LIN transceiver detecting a dominant level on the bus longer than 150 ms will detect it as a wakeup request. The LIN controller's slave is ready to listen to the bus in less than 100 ms ($T_{INITIALIZE}$<100ms) after a dominant-to-recessive edge (end-of-wakeup signal).

22.9.3 Wakeup Timeouts

The LIN protocol defines the following timeouts for a wakeup sequence. After a wakeup signal has been sent to the bus, all nodes wait for the master to send a header. If no synch field is detected before 150 ms (3,000 cycles at 20 kHz) after wakeup signal is transmitted, a new wakeup is sent by the same node that requested the first wakeup. This sequence is not repeated more than two times. After three attempts to wake up the LIN bus, wakeup signal generation is suspended for a 1.5 s (30,000 cycles at 20 kHz) period after three breaks.

NOTE: To achieve compatibility to LIN1.3 timeout conditions, the MBRS register must be set to assure that the LIN 2.0 (real-time-based) timings meet the LIN 1.3 bit time base. A node triggering the wakeup should set the MBRS register accordingly to meet the targeted time as 128 Tbits × programmed prescaler.

The LIN controller handles the wakeup expiration times defined by the LIN protocol with a hardware implementation.

22.10 Emulation Mode

In emulation mode, the CONT bit determines how the SCI/LIN operates when the program is suspended. The SCI/LIN counters are affected by this bit during debug mode. when set, the counters are not stopped and when cleared, the counters are stopped debug mode.

Any reads in emulation mode to a SCI/LIN register will not have any effect on the flags in the SCIFLR register.

NOTE: When emulation mode is entered during the Frame transmission or reception of the frame and CONT bit is not set, Communication is not expected to be successful. The suggested usage is to set CONT bit during emulation mode for successful communication.

22.11 SCI/LIN Control Registers

The SCI/LIN module registers are based on the SCI registers, with added functionality registers enabled by the LIN MODE bit in the SCIGCR1 register.

These registers are accessible in 8-, 16-, and 32-bit reads or writes. The SCI/LIN is controlled and accessed through the registers listed in Table 22-9. Among the features that can be programmed are the LIN protocol mode, communication and timing modes, baud rate value, frame format, and interrupt configuration. The base address for the control registers is FFF7 E400h.

Table 22-9. SCI/LIN Control Registers

Offset	Acronym	Register Description	Section
00h	SCIGCR0	SCI Global Control Register 0	Section 22.11.1
04h	SCIGCR1	SCI Global Control Register 1	Section 22.11.2
08h	SCIGCR2	SCI Global Control Register 2	Section 22.11.3
0Ch	SCISETINT	SCI Set Interrupt Register	Section 22.11.4
10h	SCICLEARINT	SCI Clear Interrupt Register	Section 22.11.5
14h	SCISETINTLVL	SCI Set Interrupt Level Register	Section 22.11.6
18h	SCICLEARINTLVL	SCI Clear Interrupt Level Register	Section 22.11.7
1Ch	SCIFLR	SCI Flags Register	Section 22.11.8
20h	SCIINTVECT0	SCI Interrupt Vector Offset 0	Section 22.11.9
24h	SCIINTVECT1	SCI Interrupt Vector Offset 1	Section 22.11.10
28h	SCIFORMAT	SCI Format Control Register	Section 22.11.11
2Ch	BRS	Baud Rate Selection Register	Section 22.11.12
30h	SCIED	Receiver Emulation Data Buffer	Section 22.11.13.1
34h	SCIRD	Receiver Data Buffer	Section 22.11.13.2
38h	SCITD	Transmit Data Buffer	Section 22.11.13.3
3Ch	SCIPIO0	SCI Pin I/O Control Register 0	Section 22.11.14
40h	SCIPIO1	SCI Pin I/O Control Register 1	Section 22.11.15
44h	SCIPIO2	SCI Pin I/O Control Register 2	Section 22.11.16
48h	SCIPIO3	SCI Pin I/O Control Register 3	Section 22.11.17
4Ch	SCIPIO4	SCI Pin I/O Control Register 4	Section 22.11.18
50h	SCIPIO5	SCI Pin I/O Control Register 5	Section 22.11.19
54h	SCIPIO6	SCI Pin I/O Control Register 6	Section 22.11.20
58h	SCIPIO7	SCI Pin I/O Control Register 7	Section 22.11.21
5Ch	SCIPIO8	SCI Pin I/O Control Register 8	Section 22.11.22
60h	LINCOMPARE	LIN Compare Register	Section 22.11.23
64h	LINRD0	LIN Receive Buffer 0 Register	Section 22.11.24
68h	LINRD1	LIN Receive Buffer 1 Register	Section 22.11.25
6Ch	LINMASK	LIN Mask Register	Section 22.11.26
70h	LINID	LIN Identification Register	Section 22.11.27
74h	LINTD0	LIN Transmit Buffer 0	Section 22.11.28
78h	LINTD1	LIN Transmit Buffer 1	Section 22.11.29
7Ch	MBRS	Maximum Baud Rate Selection Register	Section 22.11.30
90h	IODFTCTRL	Input/Output Error Enable Register	Section 22.11.31

22.11.1 SCI Global Control Register 0 (SCIGCR0)

The SCIGCR0 register defines the module reset. Figure 22-27 and Table 22-10 illustrate this register.

Figure 22-27. SCI Global Control Register 0 (SCIGCR0) [offset = 00]

31	16
Reserved	
R-0	

15	1	0
Reserved		RESET
R-0		RWP-0

LEGEND: R/W = Read/Write; R = Read only; RWP = Read/Write in privileged mode only; -*n* = value after reset

Table 22-10. SCI Global Control Register 0 (SCIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RESET		This bit resets the SCI/LIN module. This bit is effective in SCI and LIN mode.
		0	SCI/LIN module is in reset.
		1	SCI/LIN module is out of reset.
			Note: Read/Write in privileged mode only.

22.11.2 SCI Global Control Register 1 (SCIGCR1)

The SCIGCR1 register defines the frame format, protocol, and communication mode used by the SCI. Figure 22-28 and Table 22-11 illustrate this register.

Figure 22-28. SCI Global Control Register 1 (SCIGCR1) [offset = 04h]

31						26	25	24
Reserved							TXENA	RXENA
R-0							R/W-0	R/W-0
23						18	17	16
Reserved							CONT	LOOP BACK
R-0							R/W-0	R/W-0
15	14	13	12	11	10	9	8	
Reserved		STOP EXT FRAME	HGEN CTRL	CTYPE	MBUF MODE	ADAPT	SLEEP	
R-0		R/WL-0	R/WL-0	R/WL-0	R/W-0	R/WL-0	R/W-0	
7	6	5	4	3	2	1	0	
SWnRST	LIN MODE	CLOCK	STOP	PARITY	PARITY ENA	TIMING MODE	COMM MODE	
R/W-0	R/W-0	R/W-0	R/WC-0	R/WC-0	R/W-0	R/WC-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; WC = Write in SCI-compatible mode only; WL = Write in LIN mode only; -*n* = value after reset

Table 22-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reads return 0. Writes have no effect.
25	TXENA		Transmit enable. This bit is effective in LIN and SCI modes. Data is transferred from SCITD, or the TDy (with y=0, 1,...7) buffers in LIN mode to the SCITXSHF shift out register only when the TXENA bit is set.
		0	Transfers from SCITD or TDy to SCITXSHF are disabled.
		1	Transfers from SCITD or TDy to SCITXSHF are enabled.
			Note: Data written to SCITD or the transmit multi-buffer before TXENA is set is not transmitted. If TXENA is cleared while transmission is ongoing, the data previously written to SCITD is sent (including the checksum byte in LIN mode).
24	RXENA		Receive enable. This bit is effective in LIN and SCI modes. RXENA allows or prevents the transfer of data from SCIRXSHF to SCIRD or the receive multi-buffers.
		0	The receiver will not transfer data from the shift buffer to the receive buffer or multi-buffers.
		1	The receiver will transfer data from the shift buffer to the receive buffer or multi-buffers.
			Note: Clearing RXENA stops received characters from being transferred into the receive buffer or multi-buffers, prevents the RX status flags (see Table 22-12) from being updated by receive data, and inhibits both receive and error interrupts. However, the shift register continues to assemble data regardless of the state of RXENA.
			Note: If RXENA is cleared before a frame is completely received, the data from the frame is not transferred into the receive buffer.
			Note: If RXENA is set before a frame is completely received, the data from the frame is transferred into the receive buffer. If RXENA is set while SCIRXSHF is in the process of assembling a frame, the status flags are not assured to be accurate for that frame. To ensure that the status flags correctly reflect what was detected on the bus during a particular frame, RXENA should be set before the detection of that frame.
23-18	Reserved	0	Reads return 0. Writes have no effect.

Table 22-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
17	CONT		Continue on suspend. This bit is effective in LIN and SCI modes. This bit has an effect only when a program is being debugged with an emulator, and it determines how the SCI/LIN operates when the program is suspended. The SCI/LIN counters are affected by this bit: when the bit is set the counters are not stopped, when the bit is cleared the counters are stopped during debug mode.
		0	When debug mode is entered, the SCI/LIN state machine is frozen. Transmissions and LIN counters are halted and resume when debug mode is exited.
		1	When debug mode is entered, the SCI/LIN continues to operate until the current transmit and receive functions are complete.
16	LOOP BACK		Loopback bit. This bit is effective in LIN and SCI modes. The self-checking option for the SCI/LIN can be selected with this bit. If the LINITX and LINRX pins are configured with SCI/LIN functionality, then the LINTX pin is internally connected to the LINRX pin. Externally, during loop back operation, the LINTX pin outputs a high value and the LINRX pin is in a high-impedance state. If this bit value is changed while the SCI/LIN is transmitting or receiving data, errors may result.
		0	Loop back mode is disabled.
		1	Loop back mode is enabled.
15-14	Reserved	0	Reads return 0. Writes have no effect.
13	STOP EXT FRAME		Stop extended frame communication. This bit is effective in LIN mode only. This bit can be written only during extended frame communication. When the extended frame communication is stopped, this bit is cleared automatically.
		0	This bit has no effect.
		1	Extended frame communication will be stopped when current frame transmission/reception is completed.
12	HGEN CTRL		HGEN control. This bit is effective in LIN mode only. This bit controls the type of mask filtering comparison
		0	ID filtering using the ID-BYTE field in LIN Identification Register (LINID) occurs. Mask of FFh in LIN Mask Register (LINMASK) register will result in no match.
		1	ID filtering uses ID-SlaveTask BYTE (recommended). Mask of FFh in LIN Mask Register (LINMASK) register will result in ALWAYS match.
			Note: For software compatibility with future LIN modules the HGEN CTRL bit must be set to 1, the RX ID MASK must be set to FFh and the TX ID MASK must be set to FFh.
11	CTYPE		Checksum type. This bit is effective in LIN mode only. This bit controls the type of checksum to be used: classic or enhanced.
		0	Classic checksum is used.
		1	Enhanced checksum is used.
10	MBUF MODE		Multi-buffer mode. This bit is effective in LIN and SCI modes. This bit controls receive/transmit buffer usage, that is, whether the RX/TX multi-buffers are used or a single register, RD0/TD0, is used.
		0	The multi-buffer mode is disabled.
		1	The multi-buffer mode is enabled.
9	ADAPT		Adapt. This mode is effective in LIN mode only. This bit has an effect during the detection of the synch field. Two LIN protocol bit rate modes could be enabled with this bit according to the node capability file definition: automatic or select. The software and network configuration will decide which of these two modes are enabled. When this bit is cleared, the LIN 2.0 protocol fixed bit rate should be used. If the ADAPT bit is set, a SCI/LIN slave node detecting the baud rate will compare it to the prescalers in BRS register and update it if they are different. The BRS register will be updated with the new value. If this bit is not set there will be no adjustment to the BRS register.
		0	Automatic baud rate adjustment is disabled.
		1	Automatic baud rate adjustment is enabled.

Table 22-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
8	SLEEP		SCI sleep. This bit is effective in SCI mode only. In a multiprocessor configuration, this bit controls the receive sleep function. Clearing this bit brings the SCI/LIN out of sleep mode.
		0	Sleep mode is disabled.
		1	Sleep mode is enabled.
			Note: The receiver still operates when the SLEEP bit is set; however, RXRDY is updated and SCIRD is loaded with new data only when an address frame is detected. The remaining receiver status flags (see Table 22-12) are updated and an error interrupt is requested if the corresponding interrupt enable bit is set, regardless of the value of the SLEEP bit. In this way, if an error is detected on the receive data line while the SCI is asleep, software can promptly deal with the error condition.
			Note: The SLEEP bit is *not* automatically cleared when an address byte is detected.
			See Section 22.5.1 for more information on using the SLEEP bit for multiprocessor communication.
7	SWnRST		Software reset (active low). This bit is effective in LIN and SCI modes.
		0	The SCI/LIN is in its reset state; no data will be transmitted or received. Writing a 0 to this bit initializes the SCI/LIN state machines and operating flags as defined in Table 22-12 and Table 22-13. All affected logic is held in the reset state until a 1 is written to this bit.
		1	The SCI/LIN is in its ready state; transmission and reception can be done. After this bit is set to 1, the configuration of the module should not change.
			Note: The SCI/LIN should only be configured while SWnRST = 0.
6	LIN MODE		LIN mode. This bit is effective in LIN and SCI mode. This bit controls the module mode of operation.
		0	LIN mode is disabled; SCI mode is enabled.
		1	LIN mode is enabled; SCI mode is disabled.
5	CLOCK		SCI internal clock enable. The CLOCK bit determines the source of the module clock on the SCICLK pin. It also determines whether a LIN node is a slave or master.
			SCI mode
		0	The external SCICLK is the clock source.
			Note: If an external clock is selected, then the internal baud rate generator and baud rate registers are bypassed. The maximum frequency allowed for an externally sourced SCI clock is VCLK/16.
		1	The internal SCICLK is the clock source.
			LIN mode
		0	The node is in slave mode.
		1	The node is in master mode.
4	STOP		SCI number of stop bits per frame. This bit is effective in SCI mode only.
		0	One stop bit is used.
		1	Two stop bits are used.
			Note: The receiver checks for only one stop bit. However in idle-line mode, the receiver waits until the end of the second stop bit (if STOP = 1) to begin checking for an idle period.
3	PARITY		SCI parity odd/even selection. This bit is effective in SCI mode only. If the PARITY ENA bit is set, PARITY designates odd or even parity.
		0	Odd parity is used.
		1	Even parity is used.
			The parity bit is calculated based on the data bits in each frame and the address bit (in address-bit mode). The start and stop fields in the frame are not included in the parity calculation.
			For odd parity, the SCI transmits and expects to receive a value in the parity bit that makes odd the total number of bits in the frame with the value of 1.
			For even parity, the SCI transmits and expects to receive a value in the parity bit that makes even the total number of bits in the frame with the value of 1.

Table 22-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
2	PARITY ENA		Parity enable. This bit enables or disables the parity function.
			SCI or buffered SCI mode
		0	Parity is disabled; no parity bit is generated during transmission or is expected during reception.
		1	Parity is enabled. A parity bit is generated during transmission and is expected during reception.
			LIN mode
		0	ID field parity verification is disabled.
		1	ID field parity verification is enabled.
1	TIMING MODE		SCI timing mode bit. This bit is effective in SCI mode only. it selects the SCI timing mode.
		0	Synchronous timing is used.
		1	Asynchronous timing is used.
0	COMM MODE		SCI/LIN communication mode bit. In compatibility mode it selects the SCI communication mode. In LIN mode it selects length control option for ID-field bits ID4 and ID5.
			SCI mode
		0	Idle-line mode is used.
		1	Address-bit mode is used.
			LIN mode
		0	ID4 and ID5 are not used for length control.
		1	ID4 and ID5 are used for length control.

Table 22-12. SCI Receiver Status Flags

SCI Flag	Register	Bit	Value After Reset[1]
CE	SCIFLR	29	0
ISFE	SCIFLR	28	0
NRE	SCIFLR	27	0
FE	SCIFLR	26	0
OE	SCIFLR	25	0
PE	SCIFLR	24	0
RXWAKE	SCIFLR	12	0
RXRDY	SCIFLR	9	0
BUSY	SCIFLR	3	0
IDLE	SCIFLR	2	0
WAKE UP	SCIFLR	1	0
BRKDT	SCIFLR	0	0

(1) The flags are frozen with their reset value while SWnRST = 0.

Table 22-13. SCI Transmitter Status Flags

SCI Flag	Register	Bit	Value After Reset[1]
BE	SCIFLR	31	0
PBE	SCIFLR	30	0
TX WAKE	SCIFLR	10	0
TX EMPTY	SCIFLR	11	1
TXRDY	SCIFLR	8	1

(1) The flags are frozen with their reset value while SWnRST = 0.

22.11.3 SCI Global Control Register 2 (SCIGCR2)

The SCIGCR2 register is used to send or compare a checksum byte during extended frames, to generate a wakeup and for low-power mode control of the LIN module. Figure 22-29 and Table 22-14 illustrate this register.

Figure 22-29. SCI Global Control Register 2 (SCIGCR2) [offset = 08h]

31		18	17	16
Reserved			CC	SC
R-0			R/WL-0	R/WL-0

15	9	8	7	1	0
Reserved		GEN WU	Reserved		POWERDOWN
R-0		R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; -*n* = value after reset

Table 22-14. SCI Global Control Register 2 (SCIGCR2) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reads return 0. Writes have no effect.
17	CC		Compare checksum. LIN mode only. This bit is used by the receiver for extended frames to trigger a checksum compare. The user will initiate this transaction by writing a one to this bit. CC bit has to be set only after RX_RDY flag is set for the last received data. In non-multi-buffer mode, when the CC bit is set, the checksum will be compared on the byte that is expected to be the checksum byte. During multi-buffer mode, the following scenarios are associated with the CC bit: a) If the CC bit is set during the reception of the data, then the byte that is received after the reception of the programmed number of data bytes as indicated by SCIFORMAT[18:16] is treated as a checksum byte. b) If the CC bit is set during the idle period (that is, during the inter-frame space), then the immediate next byte will be treated as a checksum byte. c) CC bit will be auto cleared after the checkbyte has been received and compared. Checksum reception is not guaranteed if CC bit is write cleared by software during the checksum reception. See Section 22.6.6 for more details.
		0	No checksum compare will occur.
		1	Compare checksum on expected checksum byte.
16	SC		Send checksum byte. This bit is effective in LIN mode only. This bit is used by the transmitter with extended frames to send a checksum byte. In non-multi-buffer mode, the checksum byte will be sent after the current byte transmission. In multi-buffer mode, the checksum byte will be sent after the last byte count, indicated by the SCIFORMAT[18:16]). See Section 22.6.6 for more details. This byte will be cleared after the checksum byte has been transmitted. In non-multi-buffer mode, the checksum byte will be sent after the current byte transmission. During multi-buffer mode, the following scenarios are associated with the SC bit: a) The checkbyte will be sent after the last byte count, indicated by the SCIFORMAT[18:16]). b) Checksum will not be sent if SC is set before transmitting the very first byte(that is, during interframe space). c) SC bit will be auto cleared after the checkbyte has been transmitted. Checksum transmission is not guaranteed if SC bit is write cleared by software during the checksum transmission. See Section 22.6.6 for more details.
		0	No checksum byte will be sent.
		1	A checksum byte will be sent.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	GEN WU		Generate wakeup signal. This bit is effective in LIN mode only. This bit controls the generation of a wakeup signal, by transmitting the TDO buffer value. The LIN protocol specifies that this signal should be a dominant for T_{WUSIG}. This bit is cleared on reception of a valid synch break.
		0	No wakeup signal will be generated.
		1	The TDO buffer value will be transmitted for a wakeup signal. The bit will be cleared on a SWnRST.
7-1	Reserved	0	Reads return 0. Writes have no effect.

Table 22-14. SCI Global Control Register 2 (SCIGCR2) Field Descriptions (continued)

Bit	Field	Value	Description
0	POWERDOWN		Power down. This bit is effective in LIN or SCI mode. When this bit is set, the SCI/LIN module attempts to enter local low-power mode. If the POWERDOWN bit is set while the receiver is actively receiving data and the wakeup interrupt is disabled, then the SCI/LIN will delay entering low-power mode until the reception is completed. In LIN mode, the user may set the POWERDOWN bit after receiving a sleep command or on idle bus detection (more than 4 seconds). See Section 22.9 for more information on low-power mode.
		0	The SCI/LIN module is in normal operation.
		1	The SCI/LIN module enters local low-power mode.

22.11.4 SCI Set Interrupt Register (SCISETINT)

Figure 22-30 and Table 22-15 illustrate this register. Refer Figure 22-30 for details on when different interrupt flags get set in a frame during LIN Mode.

Figure 22-30. SCI Set Interrupt Register (SCISETINT) [offset = 0Ch]

31	30	29	28	27	26	25	24
SET BE INT	SET PBE INT	SET CE INT	SET ISFE INT	SET NRE INT	SET FE INT	SET OE INT	SET PE INT
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	14	13	12	10	9	8
Reserved		SET ID INT	Reserved		SET RX INT	SET TX INT
R-0		R/WL-0	R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
SET TOA3WUS INT	SET TOAWUS INT	Reserved	SET TIMEOUT INT	Reserved		SET WAKEUP INT	SET BRKDT INT
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 22-15. SCI Set Interrupt Register (SCISETINT) Field Descriptions

Bit	Field	Value	Description
31	SET BE INT		Set bit error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is a bit error.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
30	SET PBE INT		Set physical bus error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when a physical bus error occurs.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
29	SET CE INT		Set checksum-error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is a checksum error.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.

Table 22-15. SCI Set Interrupt Register (SCISETINT) Field Descriptions (continued)

Bit	Field	Value	Description
28	SET ISFE INT		Set inconsistent-synch-field-error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is an inconsistent synch field error.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
27	SET NRE INT		Set no-response-error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when a no-response error occurs.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
26	SET FE INT		Set framing-error interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when a framing error occurs.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
25	SET OE INT		Set overrun-error interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when an overrun error occurs.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
24	SET PE INT		Set parity interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when a parity error occurs.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
23-14	Reserved	0	Reads return 0. Writes have no effect.
13	SET ID INT		Set identification interrupt. This bit is effective in LIN mode only. This bit is set to enable an interrupt when a valid matching identifier is received. See Section 22.6.9 for more details.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
12-10	Reserved	0	Reads return 0. Writes have no effect.
9	SET RX INT		Receiver interrupt enable. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a receive interrupt after a frame has been completely received and the data is being transferred from SCIRXSHF to SCIRD.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
8	SET TX INT		Set transmitter interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a transmit interrupt as data is being transferred from SCITD to SCITXSHF and the TXRDY bit is being set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
7	SET TOA3WUS INT		Set timeout after three wakeup signals interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when a timeout occurs after three wakeup signals have been sent.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.

Table 22-15. SCI Set Interrupt Register (SCISETINT) Field Descriptions (continued)

Bit	Field	Value	Description
6	SET TOAWUS INT		Set timeout after wakeup signal interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when a timeout occurs after one wakeup signal has been sent.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
5	Reserved	0	Reads return 0. Writes have no effect.
4	SET TIMEOUT INT		Set timeout interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when there is no LIN bus activity (bus idle) for at least four seconds.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1	SET WAKEUP INT		Set wake-up interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a wakeup interrupt and thereby exit low-power mode. If enabled, the wakeup interrupt is asserted when local low-power mode is requested while the receiver is busy or if a low level is detected on the LINRX pin during low-power mode.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.
0	SET BRKDT INT		Set break-detect interrupt. This bit is effective in SCI-compatible mode only. Setting this bit enables the SCI/LIN to generate an error interrupt if a break condition is detected on the LINRX pin.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt is enabled.

22.11.5 SCI Clear Interrupt Register (SCICLEARINT)

Figure 22-31 and Table 22-16 illustrate this register. SCICLEARINT register is used to clear the enabled interrupts without accessing SCISETINT register.

Figure 22-31. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h]

31	30	29	28	27	26	25	24
CLR BE INT	CLR PBE INT	CLR CE INT	CLR ISFE INT	CLR RE INT	CLR FE INT	CLR OE INT	CLR PE INT
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	14	13	12	10	9	8
Reserved		CLR ID INT	Reserved		CLR RX INT	CLR TX INT
R-0		R/WL-0	R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
CLR TOA3WUS INT	CLR TOAWUS INT	Reserved	CLR TIMEOUT INT	Reserved		CLR WAKEUP INT	CLR BRKDT INT
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 22-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions

Bit	Field	Value	Description
31	CLR BE INT		Clear bit error interrupt. This bit is effective in LIN mode only. This bit disables the bit error interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
30	CLR PBE INT		Clear physical bus error interrupt. This bit is effective in LIN mode only. This bit disables the physical-bus error interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
29	CLR CE INT		Clear checksum-error interrupt. This bit is effective in LIN mode only. This bit disables the checksum interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
28	CLR ISFE INT		Clear inconsistent-synch-field-error (ISFE) interrupt. This bit is effective in LIN mode only. This bit disables the ISFE interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.

Table 22-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)

Bit	Field	Value	Description
27	CLR NRE INT		Clear no-response-error interrupt. This bit is effective in LIN mode only. This bit disables the NRE interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
26	CLR FE INT		Clear framing-error interrupt. This bit is effective in LIN or SCI mode. This bit disables the framing-error interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
25	CLR OE INT		Clear overrun-error interrupt. This bit is effective in LIN or SCI mode. This bit disables the SCI/LIN overrun error interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
24	CLR PE INT		Clear parity interrupt. This bit is effective in LIN or SCI mode. This bit disables the parity error interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
23-14	Reserved	0	Reads return 0. Writes have no effect.
13	CLR ID INT		Clear ID interrupt. This bit is effective in LIN mode only. This bit disables the ID interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
12-10	Reserved	0	Reads return 0. Writes have no effect.
9	CLR RX INT		Clear receiver interrupt. This bit is effective in LIN or SCI mode. This bit disables the receiver interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
8	CLR TX INT		Clear transmitter interrupt. This bit is effective in LIN or SCI mode. This bit disables the transmitter interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
7	CLR TOA3WUS INT		Clear timeout after three wakeup signals interrupt. This bit is effective in LIN mode only. This bit disables the timeout after three wakeup signals interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.

Table 22-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)

Bit	Field	Value	Description
6	CLR TOAWUS INT		Clear timeout after wakeup signal interrupt. This bit is effective in LIN mode only. This bit disables the timeout after one wakeup signal interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
5	Reserved	0	Reads return 0. Writes have no effect.
4	CLR TIMEOUT INT		Clear timeout interrupt. This bit is effective in LIN mode only. This bit disables the timeout (LIN bus idle) interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1	CLR WAKEUP INT		Clear wake-up interrupt. This bit is effective in LIN or SCI-compatible mode. This bit disables the wakeup interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.
0	CLR BRKDT INT		Clear break-detect interrupt. This bit is effective in SCI-compatible mode only. This bit disables the break-detect interrupt when set.
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: The interrupt is disabled.

22.11.6 SCI Set Interrupt Level Register (SCISETINTLVL)

Figure 22-32 and Table 22-17 illustrate this register.

Figure 22-32. SCI Set Interrupt Level Register (SCISETINTLVL) [offset = 14h]

31	30	29	28	27	26	25	24
SET BE INT LVL	SET PBE INT LVL	SET CE INT LVL	SET ISFE INT LVL	SET NRE INT LVL	SET FE INT LVL	SET OE INT LVL	SET PE INT LVL
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	14	13	12	10	9	8
Reserved		SET ID INT LVL	Reserved		SET RX INT LVL	SET TX INT LVL
R-0		R/WL-0	R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
SET TOA3WUS INT LVL	SET TOAWUS INT LVL	Reserved	SET TIMEOUT INT LVL	Reserved		SET WAKEUP INT LVL	SET BRKDT INT LVL
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; *-n* = value after reset

Table 22-17. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions

Bit	Field	Value	Description
31	SET BE INT LVL	0	Set bit error interrupt level. This bit is effective in LIN mode only. Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
30	SET PBE INT LVL	0	Set physical bus error interrupt level. This bit is effective in LIN mode only. Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
29	SET CE INT LVL	0	Set checksum-error interrupt level. This bit is effective in LIN mode only. Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
28	SET ISFE INT LVL	0	Set inconsistent-synch-field-error interrupt level. This bit is effective in LIN mode only. Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
27	SET NRE INT LVL	0	Set no-response-error interrupt level. This bit is effective in LIN mode only. Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
26	SET FE INT LVL	0	Set framing-error interrupt level. This bit is effective in LIN or SCI-compatible mode. Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.

Table 22-17. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
25	SET OE INT LVL		Set overrun-error interrupt level. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
24	SET PE INT LVL		Set parity error interrupt level. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
23-14	Reserved	0	Reads return 0. Writes have no effect.
13	SET ID INT LVL		Set ID interrupt level. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
12-10	Reserved	0	Reads return 0. Writes have no effect.
9	SET RX INT LVL		Set receiver interrupt level. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
8	SET TX INT LVL		Set transmitter interrupt level. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
7	SET TOA3WUS INT LVL		Set timeout after three wakeup signals interrupt level. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
6	SET TOAWUS INT LVL		Set timeout after wakeup signal interrupt level. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
5	Reserved	0	Reads return 0. Writes have no effect.
4	SET TIMEOUT INT LVL		Set timeout interrupt level. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1	SET WAKEUP INT LVL		Set wake-up interrupt level. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.
0	SET BRKDT INT LVL		Set break-detect interrupt level. This bit is effective in SCI-compatible mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read or write: The interrupt level is mapped to the INT1 line.

22.11.7 SCI Clear Interrupt Level Register (SCICLEARINTLVL)

Figure 22-33 and Table 22-18 illustrate this register.

Figure 22-33. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h]

31	30	29	28	27	26	25	24
CLR BE INT LVL	CLR PBE INT LVL	CLR CE INT LVL	CLR ISFE INT LVL	CLR NRE INT LVL	CLR FE INT LVL	CLR OE INT LVL	CLR PE INT LVL
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	14	13	12	10	9	8
Reserved		CLR ID INT LVL	Reserved		CLR RX INT LVL	CLR TX INT LVL
R-0		R/WL-0	R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
CLR TOA3WUS INT LVL	CLR TOAWUS INT LVL	Reserved	CLR TIMEOUT INT LVL	Reserved		CLR WAKEUP INT LVL	CLR BRKDT INT LVL
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; *-n* = value after reset

Table 22-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions

Bit	Field	Value	Description
31	CLR BE INT LVL		Clear bit error interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
30	CLR PBE INT LVL		Clear physical bus error interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
29	CLR CE INT LVL		Clear checksum-error interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
28	CLR ISFE INT LVL		Clear inconsistent-synch-field-error (ISFE) interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
27	CLR NRE INT LVL		Clear no-response-error interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.

Table 22-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
26	CLR FE INT LVL		Clear framing-error interrupt. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
25	CLR OE INT LVL		Clear overrun-error interrupt. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
24	CLR PE INT LVL		Clear parity interrupt. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
23-14	Reserved	0	Reads return 0. Writes have no effect.
13	CLR ID INT LVL		Clear ID interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
12-10	Reserved	0	Reads return 0. Writes have no effect.
9	CLR RX INT LVL		Clear receiver interrupt. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
8	CLR TX INT LVL		Clear transmitter interrupt. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
7	CLR TOA3WUS INT LVL		Clear timeout after three wakeup signals interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
6	CLR TOAWUS INT LVL		Clear timeout after wakeup signal interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line.
			Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line.
			Write: The interrupt level is mapped to the INT0 line.
5	Reserved	0	Reads return 0. Writes have no effect.

Table 22-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
4	CLR TIMEOUT INT LVL		Clear timeout interrupt. This bit is effective in LIN mode only.
		0	Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line. Write: The interrupt level is mapped to the INT0 line.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1	CLR WAKEUP INT LVL		Clear wake-up interrupt. This bit is effective in LIN or SCI-compatible mode.
		0	Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line. Write: The interrupt level is mapped to the INT0 line.
0	CLR BRKDT INT LVL		Clear break-detect interrupt. This bit is effective in SCI-compatible mode only.
		0	Read: The interrupt level is mapped to the INT0 line. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt level is mapped to the INT1 line. Write: The interrupt level is mapped to the INT0 line.

22.11.8 SCI Flags Register (SCIFLR)

Figure 22-34 and Table 22-19 illustrate this register.

Figure 22-34. SCI Flags Register (SCIFLR) [offset = 1Ch]

31	30	29	28	27	26	25	24
BE	PBE	CE	ISFE	NRE	FE	OE	PE
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0
23							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	ID RX	ID TX	RX WAKE	TX EMPTY	TX WAKE	RX RDY	TX RDY
R-0	R/WL-0	R/WL-0	R/WC-0	R/W-1	R/WC-0	R/W-0	R/W-1
7	6	5	4	3	2	1	0
TOA3WUS	TOAWUS	Reserved	TIMEOUT	BUSY	IDLE	WAKEUP	BRKDT
R/WL-0	R/WL-0	R-0	R/WL-0	R/W-0	R-0	R/WL-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in SCI-compatible mode only; WL = Write in LIN mode only; -*n* = value after reset

Table 22-19. SCI Flags Register (SCIFLR) Field Descriptions

Bit	Field	Value	Description
31	BE		Bit error flag. This bit is effective in LIN mode only. This bit is set when a bit error has occurred. This is detected by the internal bit monitor. See Section 22.6.8 for more information. The bit error flag is cleared by any of the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • On reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No error has been detected since this bit was last cleared. Write: Writing a 0 to this bit has no effect.
		1	Read: An error has been detected since this bit was last cleared. Write: The bit is cleared to 0.
30	PBE		Physical bus error flag. This bit is effective in LIN mode only. This bit is set when a physical bus error has been detected by the bit monitor in TED. See Section 22.6.8 for more information. The physical bus error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • On reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1 **Note: The PBE will only be flagged, if no Synch Break can be generated (for example, because of a bus shortage to VBAT) or if no Synch Break Delimiter can be generated (for example, because of a bus shortage to GND).**
		0	Read: No error has been detected since this bit was last cleared. Write: Writing a 0 to this bit has no effect.
		1	Read: An error has been detected since this bit was last cleared. Write: The bit is cleared to 0.

Table 22-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
29	CE		Checksum error flag. This bit is effective in LIN mode only. This bit is set when a checksum error has been detected by a receiving node. This error is detected by the TED logic. See Section 22.6.8 for more information. The type of checksum to be used depends on the CTYPE bit in SCIGCR1. The checksum error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No error has been detected since this bit was last cleared. Write: Writing a 0 to this bit has no effect.
		1	Read: An error has been detected since this bit was last cleared. Write: The bit is cleared to 0.
28	ISFE		Inconsistent synch field error flag. This bit is effective in LIN mode only. This bit is set when an inconsistent synch field error has been detected by the synchronizer during header reception. See Section 22.6.5.2 for more information. The inconsistent synch field error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No inconsistent synch field error has been detected. Write: Writing a 0 to this bit has no effect.
		1	Read: An inconsistent synch field error has been detected. Write: The bit is cleared to 0.
27	NRE		No-response error flag. This bit is effective in LIN mode only. This bit is set when there is no response to a master's header completed within TFRAME_MAX. This timeout period is applied for message frames of known length (identifiers 0 to 61). This error is detected by the synchronizer. See Section 22.6.7 for more information. The no-response error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No no-response error has been detected since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: A no-response error has been detected. Write: The bit is cleared to 0.

Table 22-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
26	FE		Framing error flag. This bit is effective in LIN or SCI-compatible mode. This bit is set when an expected stop bit is not found. In SCI compatibility mode, only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. Detection of a framing error causes the SCI/LIN to generate an error interrupt if the SET FE INT bit is set in the register SCISETINT. The framing error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 • Reception of a new character/frame, depending on whether the module is in SCI compatible or LIN mode In multi-buffer mode the frame is defined in the SCIFORMAT register.
		0	Read: No framing error has been detected since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: A framing error has been detected since the last clear. Write: The bit is cleared to 0.
25	OE		Overrun error flag. This bit is effective in LIN or SCI-compatible mode. This bit is set when the transfer of data from SCIRXSHF to SCIRD overwrites unread data already in SCIRD or the RDy buffers in LINRD0 and LINRD1. Detection of an overrun error causes the LIN to generate an error interrupt if the SET OE INT bit = 1. The OE flag is reset by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No overrun error has been detected since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: An overrun error has been detected. Write: The bit is cleared to 0.
24	PE		Parity error flag. This bit is effective in LIN or SCI-compatible mode. This bit is set when a parity error is detected in the received data. In SCI address-bit mode, the parity is calculated on the data and address bit fields of the received frame. In idle-line mode, only the data is used to calculate parity. An error is generated when a character is received with a mismatch between the number of 1s and its parity bit. If the parity function is disabled (SCIGCR[2] = 0), the PE flag is disabled and read as 0. Detection of a parity error causes the LIN to generate an error interrupt if the SET PE INT bit = 1. The PE bit is reset by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new character or frame, depending on whether the module is in SCI compatible or LIN mode, respectively. • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No parity error has been detected since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: A parity error has been detected. Write: The bit is cleared to 0.
23-15	Reserved	0	Reads return 0. Writes have no effect.

Table 22-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
14	ID RX		Identifier on receive flag. This bit is effective in LIN mode only. This flag is set once an identifier is received with an receive match and no ID-parity error. See Section 22.6.9 for more details. This flag indicates that a new valid identifier has been received on an RX match. This bit is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the LINID register • Reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No valid ID has been received since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: A valid ID RX has been received in LINID[23:16] on an RX match. Write: The bit is cleared to 0.
13	ID TX		Identifier on transmit flag. This bit is effective in LIN mode only. This flag is set when an identifier is received with a transmit match and no ID-parity error. See Section 22.6.9 for more details. This flag indicates that a new valid identifier has been received on a TX match. This bit is cleared by the following: • Setting the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the LINID register • Receiving a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No valid ID has been received since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: A valid ID TX has been received in LINID[23:16] on an TX match. Write: The bit is cleared to 0.
12	RX WAKE		Receiver wakeup detect flag. This bit is effective in SCI-compatible mode only. The SCI sets this bit to indicate that the data currently in SCIRD is an address. RX WAKE is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Upon receipt of a data frame.
		0	The data in SCIRD is not an address.
		1	The data in SCIRD is an address.
11	TX EMPTY		Transmitter empty flag. This flag indicates the transmitter's buffer register(s) (SCITD/TDy) and shift register (SCITXSHF) are empty. In multi-buffer mode, this flag indicates the TDx registers and shift register (SCITXSHF) are empty. In non-multi-buffer mode, this flag indicates the LINTD0 byte and the shift register (SCITXSHF) are empty. **Note: The RESET bit, an active SWnRST (SCIGCR1[7]) or a system reset sets this bit. This bit does not cause an interrupt request.** *SCI mode or LIN non-multi-buffer mode*
		0	Transmitter buffer or shift register (or both) are loaded with data.
		1	Transmitter buffer and shift registers are both empty.
			In LIN mode using multi-buffer mode
		0	Multi-buffer or shift register (or all) are loaded with data.
		1	Multi-buffer and shift registers are all empty.

Table 22-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
10	TX WAKE		Transmitter wakeup method select. This bit is effective in SCI mode only. The TX WAKE bit controls whether the data in SCITD should be sent as an address or data frame using multiprocessor communication format. This bit is set to 1 or 0 by software before a byte is written to SCITD and is cleared by the SCI when data is transferred from SCITD to SCITXSHF or by a system reset. **Note: TXWAKE is not cleared by the SWnRST bit.** *Address-bit mode*
		0	Frame to be transmitted will be data (address bit = 0).
		1	Frame to be transmitted will be an address (address bit = 1).
			Idle-line mode
		0	The frame to be transmitted will be data.
		1	The following frame to be transmitted will be an address (writing a 1 to this bit followed by writing dummy data to the SCITD will result in a idle period of 11 bit periods before the next frame is transmitted).
9	RX RDY		Receiver ready flag. In SCI-compatible mode, the receiver sets this bit to indicate that the SCIRD contains new data and is ready to be read by the CPU. In *LIN mode,* RX RDY is set once a valid frame is received in multi-buffer mode, a valid frame being a message frame received with no errors. In *non-multi-buffer mode,* RX RDY is set for each received byte and will be set for the last byte of the frame if there are no errors. The SCI/LIN generates a receive interrupt when RX RDY flag bit is set if the SET RX INT bit is set (SCISETINT[9]); RX RDY is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the SCIRD register in compatibility mode • Reading the last data byte RDy of the response in LIN mode **Note: The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.**
		0	Read: No new data is in SCIRD. Write: Writing a 0 to this bit has no effect.
		1	Read: New data is ready to be read from SCIRD. Write: The bit is cleared to 0.
8	TX RDY		Transmitter buffer register ready flag. When set, this bit indicates that the transmit buffer(s) register(s) (SCITD in compatibility mode and LINTD0/LINTD1 in multi-buffer mode) are ready to get another character from a CPU write. *In SCI,* writing data to SCITD automatically clears this bit. In *LIN mode,* this bit is cleared once byte 0 (TD0) is written to LINTD0. This bit is set after the data of the TX buffer is shifted into the SCITXSHF register. This event can trigger a transmit interrupt after data is copied to the TX shift register SCITXSHF, if the SET TX INT is set. **Note:** **1) TXRDY is also set to 1 either by setting of the RESET bit, enabling SWnRST or by a system reset.** **2) The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.** **3) The transmit interrupt request can be eliminated until the next series of data written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLEARINT register or by disabling the transmitter via the TXENA bit.** *SCI mode*
		0	SCITD is full.
		1	SCITD is ready to receive the next character.
			LIN mode
		0	The multi-buffers are full.
		1	The multi-buffers are ready to receive the next character(s). For more information on transmit interrupt handling, see the SCI document for compatibility mode and Section 22.6.9 for LIN mode.

Table 22-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
7	TOA3WUS		Timeout after three wakeup signals flag. This bit is effective in LIN mode only. This flag is set if there is no synch break received after three wakeup signals and a period of 1.5 seconds has passed. Such expiration time is used before issuing another round of wakeup signals. This bit is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 See Section 22.9.3 for more information.
		0	Read: No timeout occurred after three wakeup signals. Write: Writing a 0 to this bit has no effect.
		1	Read: Timeout occurred after three wakeup signals and 1.5 seconds time. Write: The bit is cleared to 0.
6	TOAWUS		Timeout after wakeup signal flag. This bit is effective in LIN mode only. This bit is set if there is no synch break received after a wakeup signal has been sent. A minimum of 150 ms expiration time is used before issuing another wakeup signal. This bit is cleared by the following: • Setting the SWnRST bit • Setting of the RESET bit • A system reset occurring • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 See Section 22.9.3 for more information.
		0	Read: No timeout occurred after one wakeup signal (150 ms). Write: Writing a 0 to this bit has no effect.
		1	Read: Timeout occurred after one wakeup signal. Write: The bit is cleared to 0.
5	Reserved	0	Reads return 0. Writes have no effect.
4	TIMEOUT		LIN bus idle timeout flag. This bit is effective in LIN mode only. This bit is set earliest after at least four seconds of bus inactivity. Bus inactivity is defined as no transactions between recessive and dominant (and vice versa). This bit is cleared by the following: • Setting the SWnRST bit • Setting of the RESET bit • A system reset occurring • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 See Section 22.6.7 for more information.
		0	Read: No bus idle has been detected since this bit was last cleared. Write: Writing a 0 to this bit has no effect.
		1	Read: A LIN bus idle has been detected. Write: The bit is cleared to 0.
3	BUSY		Bus busy flag. This bit is effective in LIN mode and SCI-compatible mode. This bit indicates whether the receiver is in the process of receiving a frame. As soon as the receiver detects the beginning of a start bit, the BUSY bit is set to 1. When the reception of a frame is complete, the SCI/LIN clears the BUSY bit. If SET WAKEUP INT is set and power down is requested while this bit is set, the SCI/LIN automatically prevents low-power mode from being entered and generates wakeup interrupt. The BUSY bit is controlled directly by the SCI/LIN receiver, but this bit can also be cleared by the following: • Setting the SWnRST bit • Setting of the RESET bit • A system reset occurring
		0	The receiver is not currently receiving a frame.
		1	The receiver is currently receiving a frame.

Table 22-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
2	IDLE		SCI receiver in idle state. This bit is effective in SCI-compatible mode only. While this bit is set, the SCI looks for an idle period to resynchronize itself with the bit stream. The receiver does not receive any data while the bit is set. The bus must be idle for 11 bit periods to clear this bit. The SCI enters the idle state if one of the following events occurs: • A system reset • An SCI software reset • A power down • The RX pin is configured as a general I/O pin
		0	The idle period has been detected; the SCI is ready to receive.
		1	The idle period has not been detected; the SCI will not receive any data.
1	WAKEUP		Wakeup flag. This bit is effective in LIN mode only. This bit is set by the SCI/LIN when receiver or transmitter activity has taken the module out of power-down mode. An interrupt is generated if the SET WAKEUP INT bit (SCISETINT[2]) is set. It is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 For compatibility mode, see the SCI document for more information on low-power mode.
		0	Read: The module will not wake up from power-down mode. Write: Writing a 0 to this bit has no effect.
		1	Read: Wake up from power-down mode. Write: The bit is cleared to 0.
0	BRKDT		SCI break-detect flag. This bit is effective in SCI-compatible mode only. This bit is set when the SCI detects a break condition on the LINRX pin. A break condition occurs when the LINRX pin remains continuously low for at least 10 bits after a missing first stop bit, that is, after a framing error. Detection of a break condition causes the SCI to generate an error interrupt if the SET BRKDT INT bit is set. The BRKDT bit is reset by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	Read: No break condition has been detected since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: A break condition has been detected. Write: The bit is cleared to 0.

22.11.9 SCI Interrupt Vector Offset 0 (SCIINTVECT0)

Figure 22-35 and Table 22-20 illustrate this register.

Figure 22-35. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h]

31	16
Reserved	
R-0	

15	5	4	0
Reserved		INTVECT0	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 22-20. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	INVECT0	0-1Fh	Interrupt vector offset for INT0. This register indicates the offset for interrupt line INT0. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag in SCIFLR corresponding to the offset that was read. See Table 22-4 for a list of the interrupts. **Note: The flags for the receive (SCIFLR[9]) and the transmit (SCIFLR[8]) interrupt cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).**

22.11.10 SCI Interrupt Vector Offset 1 (SCIINTVECT1)

Figure 22-36 and Table 22-21 illustrate this register.

Figure 22-36. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h]

31	16
Reserved	
R-0	

15	5	4	0
Reserved		INTVECT1	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 22-21. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	INVECT1	0-1Fh	Interrupt vector offset for INT1. This register indicates the offset for interrupt line INT1. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag in SCIFLR corresponding to the offset that was read. See Table 22-4 for list of interrupts. **Note: The flags for the receive (SCIFLR[9]) and the transmit (SCIFLR[8]) interrupt cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).**

22.11.11 SCI Format Control Register (SCIFORMAT)

Figure 22-37 and Table 22-22 illustrate this register.

Figure 22-37. SCI Format Control Register (SCIFORMAT) [offset = 28h]

31	19	18	16
Reserved		LENGTH	
R-0		R/W-0	

15	3	2	0
Reserved		CHAR	
R-0		R/WC-0	

LEGEND: R/W = Read/Write; R = Read only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 22-22. SCI Format Control Register (SCIFORMAT) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18-16	LENGTH	0-3h	Frame length control bits. In *LIN mode*, these bits indicate the number of bytes in the response field from 1 to 8 bytes. In *buffered SCI mode*, these bits indicate the number of characters, with the number of bits per character specified in CHAR (SCIFORMAT[2:0]). When these bits are used to indicate LIN response length (SCIGCR1[0] = 1), then when there is an ID RX match, this value should be updated with the expected length of the response. In buffered SCI mode, these bits indicate the transmitter/receiver format for the number of characters: 1 to 8. There can be up to eight characters with eight bits each.
		0	The response field has 1 byte/character.
		1h	The response field has 2 bytes/characters.
		2h	The response field has 3 bytes/characters.
		3h	The response field has 4 bytes/characters.
		4h	The response field has 5 bytes/characters.
		5h	The response field has 6 bytes/characters.
		6h	The response field has 7 bytes/characters.
		7h	The response field has 8 bytes/characters.
15-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	CHAR	0-7h	Character length control bits. These bits are effective in non-buffered SCI and buffered SCI modes only. These bits set the SCI character length from 1 to 8 bits. **In non-buffered SCI and buffered SCI mode, when data of fewer than eight bits in length is received, it is left justified in SCIRD/RDy and padded with trailing zeros. Data read from the SCIRD should be shifted by software to make the received data right justified.** **Data written to the SCITD should be right justified but does not need to be padded with leading zeros.**
		0	The character is 1 bit long.
		1h	The character is 2 bits long.
		2h	The character is 3 bits long.
		3h	The character is 4 bits long.
		4h	The character is 5 bits long.
		5h	The character is 6 bits long.
		6h	The character is 7 bits long.
		7h	The character is 8 bits long.

22.11.12 Baud Rate Selection Register (BRS)

This section describes the baud rate selection register. Figure 22-38 and Table 22-23 illustrate this register.

Figure 22-38. Baud Rate Selection Register (BRS) [offset = 2Ch]

31	30 28	27 24	23 16
Rsvd	U	M	PRESCALER P
R-0	R/W-0	R/W-0	R/W-0

15 0
PRESCALER P
R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-23. Baud Rate Selection Register (BRS) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reads return 0. Writes have no effect.
30-28	U	0-2h	SCI/LIN super fractional divider selection. These bits are effective in LIN or SCI asynchronous mode. These bits are an additional fractional part for the baud rate specification. These bits allow a super-fine tuning of the fractional baud rate with seven more intermediate values for each of the M fractional divider values. See Section 22.6.4.1 for more details.
27-24	M	0-3h	SCI/LIN 4-bit fractional divider selection. These bits are effective in LIN or SCI asynchronous mode. These bits are used to select a baud rate for the SCI/LIN module, and they are a fractional part for the baud rate specification. The M divider allows fine-tuning of the baud rate over the P prescaler with 15 additional intermediate values for each of the P integer values.See Section 22.6.4.1 for more details.
23-0	PRESCALER P	0-FF FFFFh	These bits are used to select a baud rate for the SCI/LIN module. These bits are effective in LIN mode and SCI compatibility. The SCI/LIN has an internally generated serial clock determined by the VCLK and the prescalers P and M in this register. The LIN uses the 24-bit integer prescaler P value of this register to select one of over 16,700,000. The additional 4-bit fractional divider M refines the baudrate selection PRESCALER[27:24]. **NOTE: In LIN mode, ONLY the asynchronous mode and baurdrate values are used.** The baud rate can be calculated using the following formulas: $\text{Asynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{16\left(P + 1 + \frac{M}{16}\right)}\right)$ (50) $\text{Isosynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{P + 1}\right)$ (51) For P = 0, $\text{Asynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{32}\right)$ (52) $\text{Isosynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{2}\right)$ (53) Table 22-24 contains comparative baud values for different P values, with VCLK = 50 MHz, for asynchronous mode.

Table 22-24. Comparative Baud Values for Different P Values, Asynchronous Mode[1][2]

24-Bit Register Value		Baud Selected		Percent Error
Decimal	Hex	Ideal	Actual	
26	00001A	115200	115740	0.47
53	000035	57600	57870	0.47
80	000050	38400	38580	0.47
162	0000A2	19200	19172	-0.15
299	00012B	10400	10417	0.16
325	000145	9600	9586	-0.15
399	00018F	7812.5	7812.5	0.00
650	00028A	4800	4800	0.00
15624	003BA0	200	200	0.00
624999	098967	5	5	0.00

(1) VCLK = 50 MHz
(2) Values are in decimal except for column 2.

22.11.13 SCI Data Buffers (SCIED, SCIRD, SCITD)

The SCI has three addressable registers in which transmit and receive data is stored. These three registers are available in SCI mode only.

22.11.13.1 Receiver Emulation Data Buffer (SCIED)

The SCIED register is addressed at a location different from SCIRD, but is physically the same register. Figure 22-39 and Table 22-25 illustrate this register.

Figure 22-39. Receiver Emulation Data Buffer (SCIED) [offset = 30h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		ED	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 22-25. Receiver Emulation Data Buffer (SCIED) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	ED	0-FFh	Emulator data. This bit is effective in SCI-compatible mode only. Reading SCIED[7:0] does not clear the RXRDY flag, unlike reading SCIRD. This register should be used only by an emulator that must continually read the data buffer without affecting the RXRDY flag.

22.11.13.2 Receiver Data Buffer (SCIRD)

This register provides a location for the receiver data. Figure 22-40 and Table 22-26 illustrate this register.

Figure 22-40. Receiver Data Buffer (SCIRD) [offset = 34h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RD	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 22-26. Receiver Data Buffer (SCIRD) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	RD	0-FFh	Receiver data. This bit is effective in SCI-compatible mode only. When a frame has been completely received, the data in the frame is transferred from the receiver shift register SCIRXSHF to this register. As this transfer occurs, the RXRDY flag is set and a receive interrupt is generated if SET RX INT is set. **Note: When the data is read from SCIRD, the RXRDY flag is automatically cleared.**

NOTE: When the SCI receives data that is fewer than eight bits in length, it loads the data into this register in a left justified format padded with trailing zeros. Therefore, the user software should perform a logical shift on the data by the correct number of positions to make it right justified.

22.11.13.3 Transmit Data Buffer Register (SCITD)

Data to be transmitted is written to the SCITD register. Figure 22-41 and Table 22-27 illustrate this register.

Figure 22-41. Transmit Data Buffer Register (SCITD) [offset = 38h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		TD	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-27. Transmit Data Buffer Register (SCITD) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	TD	0-FFh	Transmit data. This bit is effective in SCI-compatible mode only. Data to be transmitted is written to the SCITD register. The transfer of data from this register to the transmit shift register SCITXSHF sets the TXRDY flag, which indicates that SCITD is ready to be loaded with another byte of data. **Note: If SET TX INT is set, this data transfer also causes an interrupt.**

NOTE: Data written to the SCITD register that is fewer than eight bits long must be right justified, but it does not need to be padded with leading zeros.

22.11.14 SCI Pin I/O Control Register 0 (SCIPIO0)

Figure 22-42 and Table 22-28 illustrate this register.

Figure 22-42. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX FUNC	RX FUNC	Reserved
R-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-28. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX FUNC		Transfer function. This bit is effective in LIN or SCI mode. This bit defines the function of pin LINTX.
		0	LINTX is a general-purpose digital I/O pin.
		1	LINTX is the SCI/LIN transmit pin.
1	RX FUNC		Receive function. This bit is effective in LIN or SCI mode. This bit defines the function of pin LINRX.
		0	LINRX is a general-purpose digital I/O pin.
		1	LINRX is the SCI/LIN receive pin.
0	Reserved	0	Reads return 0. Writes have no effect.

22.11.15 SCI Pin I/O Control Register 1 (SCIPIO1)

Figure 22-43 and Table 22-29 illustrate this register.

Figure 22-43. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX DIR	RX DIR	Reserved
R-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-29. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX DIR		Transmit pin direction. This bit is effective in LIN or SCI mode. This bit determines the data direction on the LINTX pin if it is configured with general-purpose I/O functionality (TX FUNC = 0). See Table 22-30 for the LINTX pin control with this bit and others.
		0	LINTX is a general-purpose input pin.
		1	LINTX is a general-purpose output pin.
1	RX DIR		Receive pin direction. This bit is effective in LIN or SCI mode. This bit determines the data direction on the LINRX pin if it is configured with general-purpose I/O functionality (RX FUNC = 0). See Table 22-31 for the LINRX pin control with this bit and others.
		0	LINRX is a general-purpose input pin.
		1	LINRX is a general-purpose output pin.
0	Reserved	0	Reads return 0. Writes have no effect.

Table 22-30. LINTX Pin Control

Function	TX IN [1]	TX OUT	TX FUNC	TX DIR
LINTX	X	X	1	X
General purpose input	X	X	0	0
General purpose output, high	X	1	0	1
General purpose output, low	X	0	0	1

(1) TX IN is a read-only bit. Its value always reflects the level of the LINTX pin.

Table 22-31. LINRX Pin Control

Function	RX IN [1]	RX OUT	RX FUNC	RX DIR
LINRX	X	X	1	X
General purpose input	X	X	0	0
General purpose output, high	X	1	0	1
General purpose output, low	X	0	0	1

(1) RX IN is a read-only bit. Its value always reflects the level of the LINRX pin.

22.11.16 SCI Pin I/O Control Register 2 (SCIPIO2)

Figure 22-44 and Table 22-32 illustrate this register.

Figure 22-44. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX IN	RX IN	Reserved
R-0		R-X	R-X	R-X

LEGEND: R = Read only; X = value is indeterminate; -*n* = value after reset

Table 22-32. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX IN		Transmit pin in. This bit is effective in LIN or SCI mode. This bit contains the current value on the LINTX pin.
		0	The LINTX pin is at logic low (0).
		1	The LINTX pin is at logic high (1).
1	RX IN		Receive pin in. This bit is effective in LIN or SCI mode. This bit contains the current value on the LINRX pin.
		0	The LINRX pin is at logic low (0).
		1	The LINRX pin is at logic high (1).
0	Reserved	x	Writes have no effect.

22.11.17 SCI Pin I/O Control Register 3 (SCIPIO3)

Figure 22-45 and Table 22-33 illustrate this register.

Figure 22-45. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX OUT	RX OUT	Reserved
R-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX OUT		Transmit pin out. This bit is effective in LIN or SCI mode. This pin specifies the logic to be output on pin LINTX if the following conditions are met: • TX FUNC = 0 (LINTX pin is a general-purpose I/O.) • TX DIR = 1 (LINTX pin is a general-purpose output.) See Table 22-30 for an explanation of this bit's effect in combination with other bits.
		0	The output on the LINTX pin is at logic low (0).
		1	The output on the LINTX pin is at logic high (1). (Output voltage is V_{OH} or higher if TXPDR = 0, and output is in high-impedance state if TXPDR = 1)
1	RX OUT		Receive pin out. This bit is effective in LIN or SCI mode. This bit specifies the logic to be output on pin LINRX if the following conditions are met: • RX FUNC = 0 (LINRX pin is a general-purpose I/O.) • RX DIR = 1 (LINRX pin is a general-purpose output.) See Table 22-31 for an explanation of this bit's effect in combination with the other bits.
		0	The output on the LINRX pin is at logic low (0).
		1	The output on the LINRX pin is at logic high (1). (Output voltage is V_{OH} or higher if RXPDR = 0, and output is in high-impedance state if RXPDR = 1)
0	Reserved	0	Reads return 0. Writes have no effect.

22.11.18 SCI Pin I/O Control Register 4 (SCIPIO4)

Figure 22-46 and Table 22-34 illustrate this register.

Figure 22-46. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX SET	RX SET	Reserved
R-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX SET		Transmit pin set. This bit is effective in LIN or SCI mode. This bit sets the logic to be output on pin LINTX if the following conditions are met: • TX FUNC = 0 (LINTX pin is a general-purpose I/O.) • TX DIR = 1 (LINTX pin is a general-purpose output.) See Table 22-30 for an explanation of this bit's effect in combination with other bits.
		0	Read: The output on LINTX pin is at logic low (0). Write: Writing a 0 to this bit has no effect.
		1	Read or write: The output on LINTX pin is at logic high (1).
1	RX SET		Receive pin set. This bit is effective in LIN or SCI mode. This bit sets the data to be output on pin LINRX if the following conditions are met: • RX FUNC = 0 (LINRX pin is a general-purpose I/O.) • RX DIR = 1 (LINRX pin is a general-purpose output.) See Table 22-31 for an explanation of this bit's effect in combination with the other bits.
		0	Read: The output on LINRX pin is at logic low (0). Write: Writing a 0 to this bit has no effect.
		1	Read or write: The output on LINRX pin is at logic high (1).
0	Reserved	0	Reads return 0. Writes have no effect.

22.11.19 SCI Pin I/O Control Register 5 (SCIPIO5)

Figure 22-47 and Table 22-35 illustrate this register.

Figure 22-47. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX CLR	RX CLR	Reserved
R-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX CLR		Transmit pin clear. This bit is effective in LIN or SCI mode. This bit clears the logic to be output on pin LINTX if the following conditions are met: • TX FUNC = 0 (LINTX pin is a general-purpose I/O.) • TX DIR = 1 (LINTX pin is a general-purpose output.)
		0	Read: The output on LINTX pin is at logic low (0). Write: Writing a 0 to this bit has no effect.
		1	Read: The output on LINTX pin is at logic high (1). Write: The output on LINTX pin is at logic low (0).
1	RX CLR		Receive pin clear. This bit is effective in LIN or SCI mode. This bit clears the logic to be output on pin LINRX if the following conditions are met: • RX FUNC = 0 (LINRX pin is a general-purpose I/O.) • RX DIR = 1 (LINRX pin is a general-purpose output.)
		0	Read: The output on LINRX pin is at logic low (0). Write: Writing a 0 to this bit has no effect.
		1	Read: The output on LINRX pin is at logic high (1). Write: The output on LINRX pin is at logic low (0).
0	Reserved	0	Reads return 0. Writes have no effect.

22.11.20 SCI Pin I/O Control Register 6 (SCIPIO6)

Figure 22-48 and Table 22-36 illustrate this register.

Figure 22-48. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h]

31	8
Reserved	
R-0	

3	2	1	0
Reserved	TX PDR	RX PDR	Reserved
R-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PDR		Transmit pin open drain enable. This bit is effective in LIN or SCI mode. This bit enables open-drain capability in the output pin LINTX, if the following conditions are met: • TX FUNC = 0 (LINTX pin is a general-purpose I/O.) • TX DIR = 1 (LINTX pin is a general-purpose output.)
		0	Open-drain functionality is disabled; the output voltage is V_{OL} or lower if TXOUT = 0, and is V_{OH} or higher if TXOUT = 1.
		1	Open-drain functionality is enabled; the output voltage is V_{OL} or lower if TXOUT = 0, and is high-impedance if TXOUT = 1.
1	RX PDR		Receive pin open drain enable. This bit is effective in LIN or SCI mode. This bit enables open-drain capability in the output pin LINRX, if the following conditions are met: • RX FUNC = 0 (LINRX pin is a general-purpose I/O.) • RX DIR = 1 (LINRX pin is a general-purpose output.)
		0	Open-drain functionality is disabled; the output voltage is V_{OL} or lower if RXOUT = 0, and is V_{OH} or higher if RXOUT = 1.
		1	Open-drain functionality is enabled; the output voltage is V_{OL} or lower if RXOUT = 0, and is high-impedance if RXOUT = 1.
0	Reserved	0	Reads return 0. Writes have no effect.

22.11.21 SCI Pin I/O Control Register 7 (SCIPIO7)

Figure 22-49 and Table 22-37 illustrate this register.

Figure 22-49. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX PD	RX PD	Reserved
R-0		R/W-n	R/W-n	R-n

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 22-37. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PD		Transmit pin pull control disable. This bit is effective in LIN or SCI mode. This bit disables pull control capability on the input pin LINTX.
		0	The pull control on the LINTX pin is enabled.
		1	The pull control on the LINTX pin is disabled.
1	RX PD		Receive pin pull control disable. This bit is effective in LIN or SCI mode. This bit disables pull control capability on the input pin LINRX.
		0	Pull control on the LINRX pin is enabled.
		1	Pull control on the LINRX pin is disabled.
0	Reserved	0	Writes have no effect.

22.11.22 SCI Pin I/O Control Register 8 (SCIPIO8)

Figure 22-50 and Table 22-38 illustrate this register.

Figure 22-50. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX PSL	RX PSL	Reserved
R-0		R/W-n	R/W-n	R-n

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 22-38. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PSL		TX pin pull select. This bit is effective in LIN or SCI mode. This bit selects pull type in the input pin LINTX.
		0	The LINTX pin is a pull down.
		1	The LINTX pin is a pull up.
1	RX PSL		RX pin pull select. This bit is effective in LIN or SCI mode. This bit selects pull type in the input pin LINRX.
		0	The LINRX pin is a pull down.
		1	The LINRX pin is a pull up.
0	Reserved	0	Writes have no effect.

22.11.23 LIN Compare Register (LINCOMPARE)

Figure 22-51 and Table 22-39 illustrate this register.

Figure 22-51. LIN Compare Register (LINCOMPARE) [offset = 60h]

31	16
Reserved	
R-0	

15	10	9	8	7	3	2	0
Reserved		SDEL		Reserved		SBREAK	
R-0		RWP-0		R-0		RWP-0	

LEGEND: R/W = Read/Write; R = Read only; RWP = Read/Write in privileged mode only; --*n* = value after reset

Table 22-39. LIN Compare Register (LINCOMPARE) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	SDEL		2-bit synch delimiter compare. These bits are effective in LIN mode only. These bits are used to configure the number of T_{bit} for the synch delimiter in the synch field. The default value is 0. The formula to program the value (in T_{bits}) for the synchronization delimiter is: $T_{SDEL} = (SDEL + 1)\ T_{bit}$
		0	The synch delimiter has 1 T_{bit}.
		1h	The synch delimiter has 2 T_{bit}.
		2h	The synch delimiter has 3 T_{bit}.
		3h	The synch delimiter has 4 T_{bit}.
7-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	SBREAK		Synch break extend. These bits are effective in LIN mode only. These bits are used to configure the number of T_{bit} for the synch break to extend the minimum 13 T_{bit} break field to a maximum of 20 T_{bit} long. Note: The default value is 0, which adds nothing to the automatically generated SYNCH BREAK. The formula to program the value (in T_{bits}) for the SYNCH BREAK is: $T_{SYNBRK} = 13T_{bit} + (SBREAK \times T_{bit})$
		0	The synch break has no additional T_{bit}.
		1h	The synch break has 1 additional T_{bit}.
		2h	The synch break has 2 additional T_{bit}.
		3h	The synch break has 3 additional T_{bit}.
		4h	The synch break has 4 additional T_{bit}.
		5h	The synch break has 5 additional T_{bit}.
		6h	The synch break has 6 additional T_{bit}.
		7h	The synch break has 7 additional T_{bit}.

22.11.24 LIN Receive Buffer 0 Register (LINRD0)

Figure 22-52 and Table 22-40 illustrate this register.

Figure 22-52. LIN Receive Buffer 0 Register (LINRD0) [offset = 64h]

31	24	23	16
RD0		RD1	
R-0		R-0	

15	8	7	0
RD2		RD3	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 22-40. LIN Receive Buffer 0 Register (LINRD0) Field Descriptions

Bit	Field	Value	Description
31-24	RD0	0-FFh	Receive buffer 0. Byte 0 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy bit field according to the number of bytes received. A read of this byte clears the RXDY byte. **Note: RD<x-1> is equivalent to data byte <x> of the LIN frame.**
23-16	RD1	0-FFh	Receive buffer 1. Byte 1 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.
15-8	RD2	0-FFh	Receive buffer 2. Byte 2 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.
7-0	RD3	0-FFh	Receive buffer 3. Byte 3 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.

22.11.25 LIN Receive Buffer 1 Register (LINRD1)

Figure 22-53 and Table 22-41 illustrate this register.

Figure 22-53. LIN Receive Buffer 1 Register (RD1) [offset = 68h]

31	24	23	16
RD4		RD5	
R-0		R-0	

15	8	7	0
RD6		RD7	
R-0		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 22-41. LIN Receive Buffer 1 Register (RD1) Field Descriptions

Bit	Field	Value	Description
31-24	RD4	0-FFh	Receive buffer 4. Byte 4 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received. **Note: RD<x-1> is equivalent to data byte <x> of the LIN frame.**
23-16	RD5	0-FFh	Receive buffer 5. Byte 5 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.
15-8	RD6	0-FFh	Receive buffer 6. Byte 6 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.
7-0	RD7	0-FFh	Receive buffer 7. Byte 7 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.

22.11.26 LIN Mask Register (LINMASK)

Figure 22-54 and Table 22-42 illustrate this register.

Figure 22-54. LIN Mask Register (LINMASK) [offset = 6Ch]

31	24	23	16
Reserved		RX ID MASK	
R-0		R/WL-0	

15	8	7	0
Reserved		TX ID MASK	
R-0		R/WL-0	

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; -*n* = value after reset

Table 22-42. LIN Mask Register (LINMASK) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	RX ID MASK	0-FFh	Receive ID mask. These bits are effective in LIN mode only. This 8-bit mask is used for filtering an incoming ID message and comparing it to the ID-byte. A compare match of the received ID with the RX ID MASK will set the ID RX flag and trigger an ID interrupt if enabled (SET ID INT in SCISETINT). A 0 bit in the mask indicates that bit is compared to the ID-byte. A 1 bit in the mask indicates that bit is filtered and therefore is not used in the compare.
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	TX ID MASK	0-FFh	Transmit ID mask. These bits are effective in LIN mode only. This 8-bit mask is used for filtering an incoming ID message and comparing it to the ID-byte. A compare match of the received ID with the TX ID MASK will set the ID TX flag and trigger an ID interrupt if enabled (SET ID INT in SCISETINT). A 0 bit in the mask indicates that bit is compared to the ID-byte. A 1 bit in the mask indicates that bit is filtered and therefore is not used for the compare.

22.11.27 LIN Identification Register (LINID)

Figure 22-55 and Table 22-43 illustrate this register.

Figure 22-55. LIN Identification Register (LINID) [offset = 70h]

31	24	23	16
Reserved		RECEIVED ID	
R-0		R-0	

15	8	7	0
ID-SLAVETASK BYTE		ID BYTE	
R/WL-0		R/WL-0	

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; -*n* = value after reset

Table 22-43. LIN Identification Register (LINID) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	RECEIVED ID	0-FFh	Received identification. These bits are effective in LIN mode only. This byte contains the current message identifier. During header reception the received ID is copied from the SCIRXSHF register to this byte if there is no ID-parity error and there has been an RX/TX match.
15-8	ID-SLAVETASK BYTE	0-FFh	ID-SlaveTask Byte. These bits are effective in LIN mode only. This field contains the identifier to which the received ID of an incoming header will be compared to decide whether a receive response, a transmit response, or no action needs to be performed by the LIN node when a header with that particular ID is received.
7-0	ID BYTE	0-FFh	ID byte. This field is effective in LIN mode only. This byte is the LIN mode message ID. On a master node, a write to this register by the CPU initiates a header transmission. For a slave task, this byte is used for message filtering when HGEN CTRL = 0.

NOTE: For software compatibility with future LIN modules, the HGEN CTRL bit must be set to 1, the RX ID MASK field must be set to FFh, and the TX ID MASK field must be set to FFh.

22.11.28 LIN Transmit Buffer 0 Register (LINTD0)

Figure 22-56 and Table 22-44 illustrate this register.

Figure 22-56. LIN Transmit Buffer 0 Register (LINTD0) [offset = 74h]

31	24	23	16
TD0		TD1	
R/W-0		R/W-0	

15	8	7	0
TD2		TD3	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 22-44. LIN Transmit Buffer 0 Register (LINTD0) Field Descriptions

Bit	Field	Value	Description
31-24	TD0	0-FFh	8-Bit transmit buffer 0. Byte 0 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TD0 buffer, transmission will be initiated. **Note: TD<x-1> is equivalent to data byte <x> of the LIN frame.**
23-16	TD1	0-FFh	8-Bit transmit buffer 1. Byte 1 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
15-8	TD2	0-FFh	8-Bit transmit buffer 2. Byte 2 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
7-0	TD3	0-FFh	8-Bit transmit buffer 3. Byte 3 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

22.11.29 LIN Transmit Buffer 1 Register (LINTD1)

Figure 22-57 and Table 22-45 illustrate this register.

Figure 22-57. LIN Transmit Buffer 1 Register (LINTD1) [offset = 78h]

31	24	23	16
TD4		TD5	
R/W-0		R/W-0	

15	8	7	0
TD6		TD7	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 22-45. LIN Transmit Buffer 1 Register (LINTD1) Field Descriptions

Bit	Field	Value	Description
31-24	TD4	0-FFh	8-Bit transmit buffer 4. Byte 4 to be transmitted is written into this register and then copied to SCITXSHF for transmission. **Note: TD<x-1> is equivalent to data byte <x> of the LIN frame.**
23-16	TD5	0-FFh	8-Bit transmit buffer 5. Byte 5 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
15-8	TD6	0-FFh	8-Bit transmit buffer 6. Byte 6 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
7-0	TD7	0-FFh	8-Bit transmit buffer 7. Byte 7 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

22.11.30 Maximum Baud Rate Selection Register (MBRS)

Figure 22-58 and Table 22-46 illustrate this register.

Figure 22-58. Maximum Baud Rate Selection Register (MBRS) [offset = 7Ch]

31	16
Reserved	
R-0	

15	13	12	0
Reserved		MBR	
R-0		R/WL-DACh	

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; -*n* = value after reset

Table 22-46. Maximum Baud Rate Selection Register (MBRS) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12-0	MBR	0-1FFFh	Maximum baud rate prescaler. This bit is effective in LIN mode only. This 13-bit prescaler is used during the synchronization phase (see Section 22.6.5.2) of a slave module if the ADAPT bit is set. In this way, a SCI/LIN slave using an automatic or select bit rate modes detects any LIN bus legal rate automatically. The MBR value should be programmed to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise, a 0x00 data byte could mistakenly be detected as a sync break. The default value for a 70MHz VCLK is 0xDAC. This MBR prescaler is used by the wake-up and idle time counters for a constant expiration time relative to a 20 kHz rate. $MBR = \frac{0.9x\ VCLK}{maxbaudrate}$ (54)

22.11.31 Input/Output Error Enable Register (IODFTCTRL)

All the bits in the IODFTCTRL register are used in IODFT (I/O design for test) mode only. Figure 22-59 and Table 22-47 illustrate this register. After the basic SCI/LIN module configuration, enable the required Error mode to be created followed by IODFT Key enable.

NOTE: 1) All the bits are used in IODFT mode only.

2) Each IODFT are expected to be checked individually.

3) ISFE Error will not be Flagged during IODFT mode.

Figure 22-59. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h]

31	30	29	28	27	26	25	24
BEN	PBEN	CEN	ISFE	Reserved	FEN	PEN	BRKDT ENA
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R-0	R/W-0	R/WC-0	R/WC-0

23	21	20	19	18	16
Reserved		PIN SAMPLE MASK		TX SHIFT	
R-0		R/W-0		R/W-0	

15	12	11	8
Reserved		IODFTENA	
R-0		R/WP-5h	

7	2	1	0
Reserved		LPB ENA	RXP ENA
R-0		R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; WP = Write in privilege mode only; *-n* = value after reset

Table 22-47. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions

Bit	Field	Value	Description
31	BEN		Bit error enable. This bit is effective in LIN mode only. This bit is used to create a bit error.
		0	No bit error is created.
		1	The bit received is ORed with 1 and passed to the bit monitor circuitry.
30	PBEN		Physical bus error enable. This bit is effective in LIN mode only. This bit is used to create a physical bus error.
		0	No error is created.
		1	The bit received during synch break field transmission is ORed with 1 and passed to the bit monitor circuitry.
29	CEN		Checksum error enable. This bit is effective in LIN mode only. This bit is used to create a checksum error.
		0	No error is created.
		1	The polarity of the CTYPE (checksum type) in the receive checksum calculator is changed so that a checksum error is occurred.
28	ISFE		Inconsistent synch field (ISF) error enable. This bit is effective in LIN mode only. This bit is used to create an ISF error.
		0	No error is created.
		1	The bit widths in the synch field are varied so that the ISF check fails and the error flag is set.
27	Reserved	0	Reads return 0. Writes have no effect.
26	FEN		Frame error enable. This bit is used to create a frame error.
		0	No error is created.
		1	The stop bit received is ANDed with 0 and passed to the stop bit check circuitry.

Table 22-47. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
25	PEN		Parity error enable. This bit is effective in SCI-compatible mode only. This bit is used to create a parity error.
		0	No parity error occurs.
		1	The parity bit received is toggled so that a parity error occurs.
24	BRKDT ENA		Break detect error enable. This bit is effective in SCI-compatible mode only. This bit is used to create a BRKDT error.
		0	No error is created.
		1	The stop bit of the frame is ANDed with 0 and passed to the RSM so that a frame error occurs. Then the RX pin is forced to continuous low for 10 T_{bit} so that a BRKDT error occurs.
32-21	Reserved	0	Reads return 0. Writes have no effect.
20-19	PIN SAMPLE MASK		Pin sample mask. These bits define the sample number at which the TX pin value that is being transmitted will be inverted to verify the receive pin samples majority detection circuitry.
			Note: In IODFT mode testing for pin_sample mask must be done with prescalar P programmed greater than 2 (P > 2).
		0	No mask is used.
		1h	Invert the TX Pin value at TBIT_CENTER.
		2h	Invert the TX Pin value at TBIT_CENTER + SCLK.
		3h	Invert the TX Pin value at TBIT_CENTER + 2 SCLK.
18-16	TX SHIFT		Transmit shift. These bits define the amount by which the value on TX pin is delayed so that the value on the RX pin is asynchronous. This feature is not applicable to the start bit.
		0	No delay occurs.
		1h	The value is delayed by 1 SCLK.
		2h	The value is delayed by 2 SCLK.
		3h	The value is delayed by 3 SCLK.
		4h	The value is delayed by 4 SCLK.
		5h	The value is delayed by 5 SCLK.
		6h	The value is delayed by 6 SCLK.
		7h	The value is delayed by 7 SCLK.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	IODFTENA		IODFT enable key. Write access permitted in Privilege mode only.
		Ah	IODFT is enabled.
		All Others	IODFT is disabled.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	LPB ENA		Module loopback enable. Write access permitted in Privilege mode only.
			Note: In analog loopback mode the complete communication path through the I/Os can be tested, whereas in digital loopback mode the I/O buffers are excluded from this path.
		0	Digital loopback is enabled.
		1	Analog loopback is enabled in module I/O DFT mode when IODFTENA = 1010.
0	RXP ENA		Module analog loopback through receive pin enable. Write access permitted in Privilege mode only. This bit defines whether the I/O buffers for the transmit or the receive pin are included in the communication path (in analog loopback mode).
		0	Analog loopback through the transmit pin is enabled.
		1	Analog loopback through the receive pin is enabled.

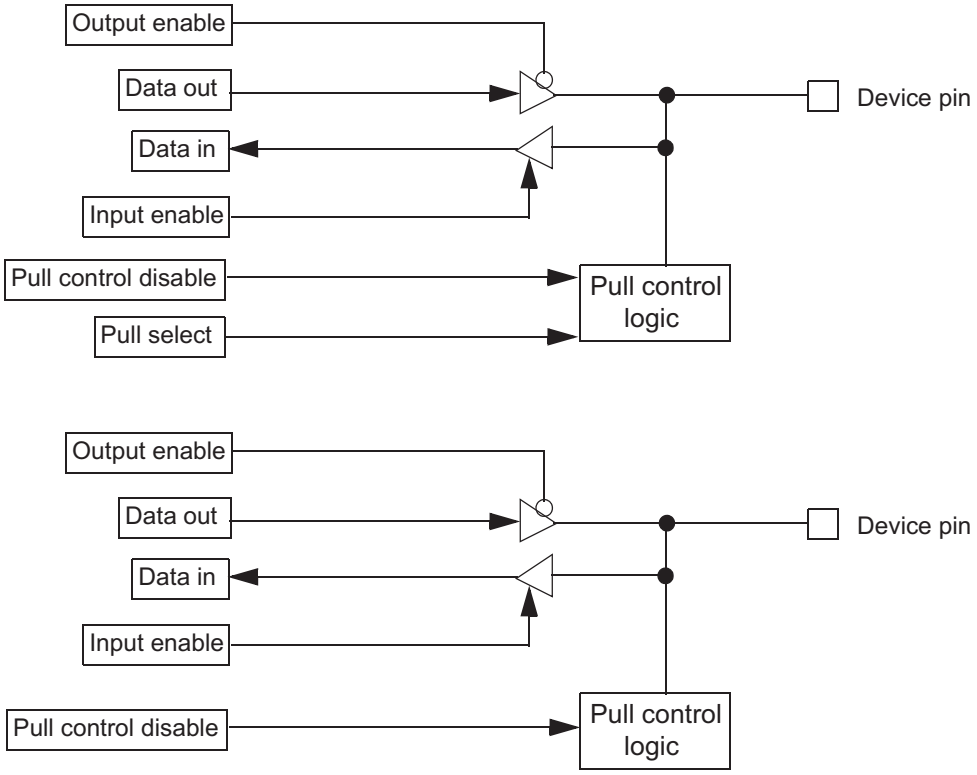
22.12 GPIO Functionality

The following sections apply to all device pins that can be configured as functional or general-purpose I/O pins.

22.12.1 GPIO Functionality

Figure 22-60 illustrates the GPIO functionality.

Figure 22-60. GPIO Functionality



22.12.2 Under Reset

The following apply if a device is under reset:

- Pull control. The reset pull control on the pins is enabled.
- Input buffer. The input buffer is enabled.
- Output buffer. The output buffer is disabled.

22.12.3 Out of Reset

The following apply if the device is out of reset:

- Pull control. The pull control is enabled by clearing the PD (pull control disable) bit in the SCIPIO7 register (Section 22.11.21). In this case, if the PSL (pull select) bit in the SCIPIO8 register (Section 22.11.22) is set, the pin will have a pull-up. If the PSL bit is cleared, the pin will have a pull-down. If the PD bit is set in the control register, there is no pull-up or pull-down on the pin.
- Input buffer. The input buffer is permanently enabled in this device.

NOTE: The pull-disable logic depends on the pin direction. It is independent of whether the device is in I/O or functional mode. If the pin is configured as output or transmit, then the pulls are disabled automatically.

- Output buffer. A pin can be driven as an output pin if the TX DIR bit is set in the pin direction control register (SCIPIO1; Section 22.11.15) AND the open-drain feature is not enabled in the SCIPIO6 register (Section 22.11.20).

22.12.4 Open-Drain Feature Enabled on a Pin

The following apply if the open-drain feature is enabled on a pin:

- The output buffer is enabled if a low signal is being driven on to the pin.
- The output buffer is disabled (the direction control signal DIR is internally forced low) if a high signal is being driven on to the pin.

NOTE: The open-drain feature is available only in I/O mode (SCIPIO0; Section 22.11.14).

22.12.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 22-48.

Table 22-48. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins

Device under Reset?	Pin Direction (DIR) [1][2]	Pull Disable (PULDIS) [1][3]	Pull Select (PULSEL) [1][4]	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Enabled	Disabled	Enabled
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Enabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

(1) X = Don't care
(2) DIR = 0 for input, = 1 for output
(3) PULDIS = 0 for enabling pull control, = 1 for disabling pull control
(4) PULSEL= 0 for pull-down functionality, = 1 for pull-up functionality

eFuse Controller

This chapter describes the eFuse controller.

Topic **Page**

23.1 Overview

Electrically programmable fuses (eFuses) are used to configure the device after deassertion of $\overline{\text{PORRST}}$. The eFuse values are read and loaded into internal registers as part of the power-on-reset sequence. The eFuse values are protected with single bit error correction, double bit error detection (SECDED) codes. These fuses are programmed during the initial factory test of the device. The eFuse controller is designed so that the state of the eFuses cannot be changed once the device is packaged.

23.2 Introduction

The eFuse controller automatically reads the values of the eFuses and shifts them into registers during the power-on reset sequence. No action is required from the application code. However, in a safety critical application, the user code should check to see if a correctable or an uncorrectable error was detected during the reset sequence and then preform a self-test on the eFuse controller ECC logic.

23.3 eFuse Controller Testing

23.3.1 eFuse Controller Connections to ESM

There are three connections from the eFuse controller to the Error Signaling Module (ESM). If an uncorrectable error occurs during the loading of the eFuse values after reset, a group three, channel one error and a group one channel 40 error are sent to the ESM. The group three error will cause the $\overline{\text{ERROR}}$ pin to go low. If during the eFuse loading a correctable error occurs, only a group one channel 40 error is sent to the ESM. If an error occurs during the eFuse controller self test, then a group one channel 41 error and a group one channel 40 error are sent to the ESM. After reset, by default, the group one errors do not affect the $\overline{\text{ERROR}}$ pin. If the software enables the appropriate bit in the appropriate ESM Influence Error Pin Set/Status Register (ESMIEPSRn) while the group one error is set, the $\overline{\text{ERROR}}$ pin will go low.

Table 23-1. ESM Signals Set by eFuse Controller

ESM Signal	Uncorrected Load Failure	Correctable Load Error	Self Test		
			eFuse Self Test	eFuse stuck at 0 Test	
				Version a: with Error pin	Version b: without Error pin
Group 3 Channel 1	X			X	
Group 1 Channel 40	X	X	X		
Group 1 Channel 41			X	X	X

23.3.2 Checking for eFuse Errors After Power Up

For safety critical systems, it is required that you check the status of the eFuse controller after a device reset. A suggested flow chart for checking the eFuse controller after device reset is shown in Figure 23-1. Failures during the eFuse self test can be grouped into three levels of severity. Depending on the safety critical application, the error handling for each error type may be different.

23.3.2.1 Class 1 Error

A class 1 error of the eFuse controller means that there was a failure during the autoload sequence. The values read from the eFuses cannot be relied on. All device operation is suspect. A class 1 error is indicated by a signal to group 3 channel 1 of the ESM. This will cause the $\overline{\text{ERROR}}$ pin to go active low.

23.3.2.2 Class 2 Errors

A class 2 error is an indication that the safety checks of the eFuse controller did not work. These are also serious errors because you can no longer guarantee that a more severe error did not occur.

23.3.2.3 Class 3 Error

A class 3 error indicates that there was a single bit failure reading the eFuses that was corrected by ECC bits. Proper operation is still likely, but the system is now at a higher risk for a future non-correctable error. When a correctable error occurs, ESM group 1, channel 40 will be set. In the suggested flow chart shown in Figure 23-1 below, the single bit error is determined by directly reading the eFuse error status register, and not depending on the integrity of the connections between the eFuse controller and the ESM.

23.3.2.4 Stuck at Zero Test

The purpose of the stuck at zero test is to verify that the eFuse controller could signal the ESM if an autoload error did occur. It basically verifies the path through the eFuse controller and to the ESM. This is done by writing a special instruction to the eFuse controller boundary register, then verifying that the proper bits are set in the eFuse controller pins register. Upon successful completion of this test ESM group 1 channel 41 and ESM group 3 channel 1 will be set. This will force the $\overline{\text{ERROR}}$ pin low.

- Version A
  - Write boundary register (address 0xFFF8C01C) with 0x003FC000 to set the error signals.
  - Read pins register (address 0xFFF8C02C) and verify that bits 14, 12, 11 and 10 are set.
  - Write boundary register (address 0xFFF8C01C) with 0x003C0000, to clear the error signals.
  - Verify that ESM group 1 channel 41 and group 3 channel 1 are set, then clear them.

If the system cannot support a test which causes the ERROR pin to go low, then the stuck at zero test can be modified as follows:

- Version B
  - Write boundary register (address 0xFFF8C01C) with 0x003BC000.
  - Read pins register (address 0xFFF8C02C) and verify that bits 14, 12, and 11 are set.
  - Write boundary register (address 0xFFF8C01C) with 0x003C0000, to clear the error signals.
  - Verify that ESM group 1 channel 41 is set, then clear it.

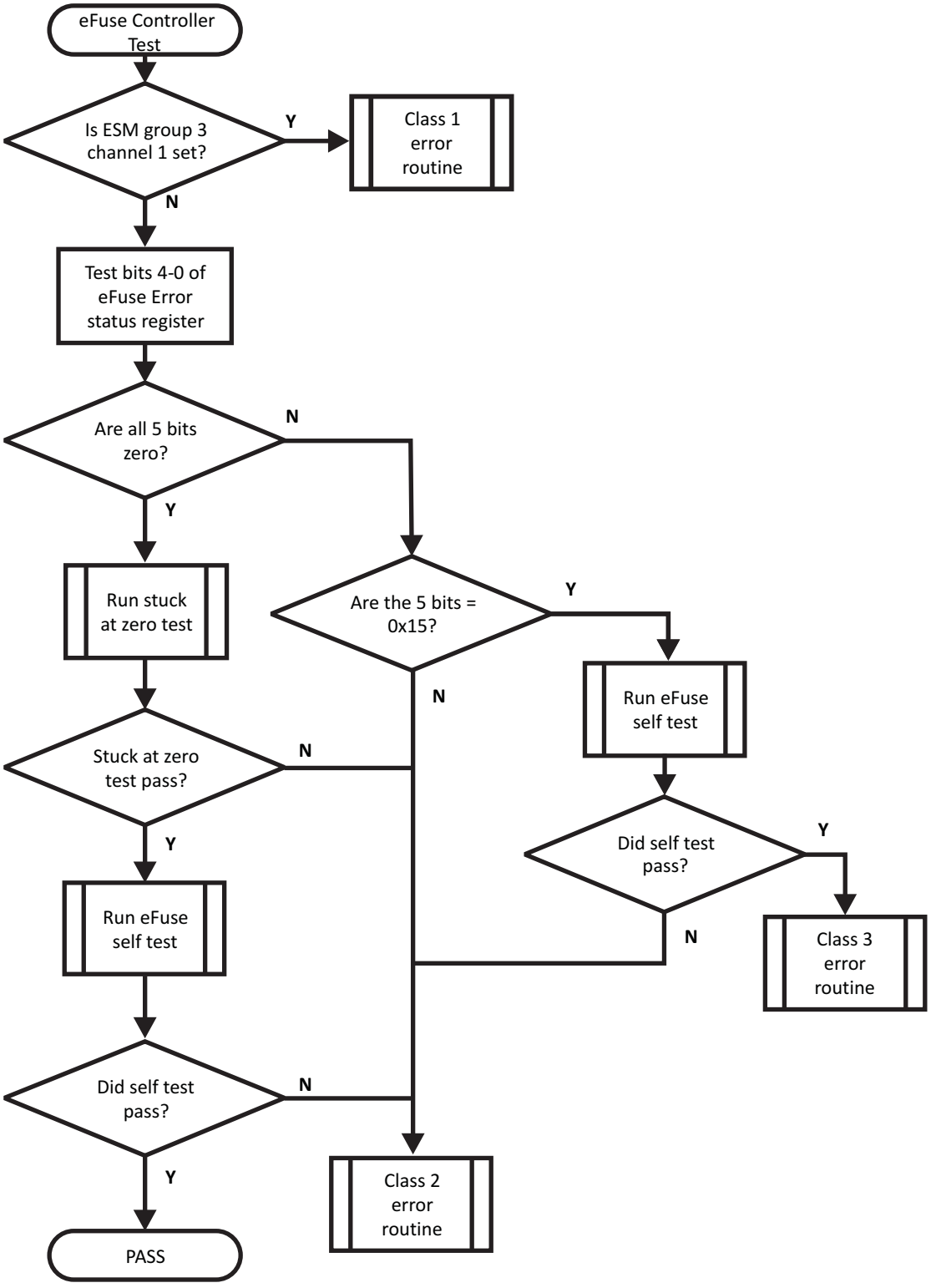
This alternate method provides less test coverage because the path from the uncorrectable error signal from the eFuse controller to the ESM is not specifically tested. However, even if this path is broken, reading the five eFuse error status bits will indicate that an error occurred.

23.3.2.5 eFuse ECC Logic Self Test

The eFuse controller self test performs extensive validation of the ECC logic in the eFuse controller. This test should only be performed once for every device $\overline{\text{PORRST}}$ cycle. Perform the self test by following these steps:

- Write 0x00000258 to the self test cycles register (EFCSTCY) at address 0xFFF8C048.
- Write 0x5362F97F to the self test signature register (EFCSTSIG) at address 0xFFF8C04C.
- Write 0x0000200F to the boundary register at address 0xFFF8C01C. This triggers the self test. The test takes 610 VCLK cycles to complete. The application can poll bit 15 of the pins register at address 0xFFF8C02C to wait for the test to complete.
- Check ESM group 1 channels 40 and 41 for any errors, neither should be set.
- Verify that bits 4 to 0 of the eFuse Error Status register at address 0xFFF8C03C are zero.

Figure 23-1. eFuse Self Test Flow Chart



23.4 eFuse Controller Registers

All registers in the eFuse Controller module are 32-bit, word-aligned; 8-bit, 16-bit and 32-bit accesses are allowed. Table 23-2 provides a quick reference to each of these registers. Specific bit descriptions are discussed in the following subsections. The base address for the control registers is FFF8 C000h.

Table 23-2. eFuse Controller Registers

Offset	Acronym	Register Description	Section
1Ch	EFCBOUND	EFC Boundary Control Register	Section 23.4.1
2Ch	EFCPINS	EFC Pins Register	Section 23.4.2
3Ch	EFCERRSTAT	EFC Error Status Register	Section 23.4.3
48h	EFCSTCY	EFC Self Test Cycles Register	Section 23.4.4
4Ch	EFCSTSIG	EFC Self Test Signature Register	Section 23.4.5

23.4.1 EFC Boundary Control Register (EFCBOUND)

Figure 23-2 and Table 23-3 describe the EFCBOUND register. The eFuse Boundary Control Register is used to test the connections between the eFuse controller and the ESM module. The eFuse Boundary Control Register is also used to initiate an eFuse controller ECC self-test.

Figure 23-2. EFC Boundary Control Register (EFCBOUND) [offset = 1Ch]

31–24	23–22	21	20	19	18	17	16
Reserved	Reserved	EFC Self Test Error	EFC Single Bit Error	EFC Instruction Error	EFC Autoload Error	Self Test Error OE	Single Bit Error OE
R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12–8	7–4	3–0
Instruction Error OE	Autoload Error OE	EFC ECC Selftest Enable	Reserved	Reserved	Input Enable
R/W-0	R/W-0	R/W-0	R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after power-on reset (nPORRST)

Table 23-3. EFC Boundary Register (EFCBOUND) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Read returns 0. Writes have no effect.
21	EFC Self Test Error		This bit drives the self test error signal when bit 17 (Self Test Error OE) is high. This signal is attached to ESM error Group 1, Channel 41.
		0	Drives the self test error signal low, if Self Test OE is high.
		1	Drives the self test error signal high, if Self Test OE is high.
20	EFC Single Bit Error		This bit drives the single bit error signal when bit 16 (Single bit Error OE) is high. This signal is attached to ESM error Group 1, Channel 40.
		0	Drives the self test error signal low, if Single Bit Error OE is high.
		1	Drives the self test error signal high, if Single Bit Error OE is high.

Table 23-3. EFC Boundary Register (EFCBOUND) Field Descriptions (continued)

Bit	Field	Value	Description
19	EFC Instruction Error		This bit drives the instruction error signal when bit 15 (Instruction Error OE) is high. This signal is used to denote an error occurred during e-fuse programming. This signal is not attached to the ESM.
		0	Drives the Instruction Error signal low, if Instruction Error OE is high.
		1	Drives the Instruction Error signal high, if Instruction Error OE is high.
18	EFC Autoload Error		This bit drives the Autoload Error signal when bit 14 (Autoload Error OE) is high. This signal is attached to ESM error Group 3, Channel 1.
		0	Drives the Autoload Error signal low, if Autoload Error OE is high.
		1	Drives the Autoload Error signal high, if Autoload Error OE is high.
17	Self Test Error OE		The Self Test Error Output Enable bit determines if the EFC Self Test signal comes from the eFuse controller or from bit 21 of the boundary register.
		0	EFC Self Test Error comes from eFuse controller.
		1	EFC Self Test Error comes from the boundary register.
16	Single Bit Error OE		The single bit error output enable signal determines if the EFC Single Bit Error signal comes from the eFuse controller or from bit 20 of the boundary register.
		0	EFC Single Bit Error comes from eFuse controller.
		1	EFC Single Bit Error comes from the boundary register.
15	Instruction Error OE		The instruction error output enable signal determines if the EFC Instruction Error signal comes from the eFuse controller or from bit 19 of the boundary register.
		0	EFC Instruction Error comes from eFuse controller.
		1	EFC Instruction Error comes from the boundary register.
14	Autoload Error OE		The autoload error output enable signal determines if the EFC Autoload Error signal comes from the eFuse controller or from bit 18 of the boundary register.
		0	EFC Autoload Error comes from eFuse controller.
		1	EFC Autoload Error comes from the boundary register.
13	EFC ECC Selftest Enable		The eFuse Controller ECC Selftest Enable bit starts the selftest of the ECC logic if the four input enable bits (EFCBOUND[3:0) are all 1s.
		0	No action
		1	Start ECC selftest if EFCBOUND[3:0] are Fh.
12-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	Input Enable		The eFuse Controller ECC Selftest Enable bit starts the selftest of the ECC logic if the four input enable bits (EFCBOUND[3:0) are all 1s.
		Fh	ECC selftest can be started if EFC ECC Selftest Enable, bit 13, is set
		All others	ECC selftest cannot be started.

23.4.2 EFC Pins Register (EFCPINS)

Figure 23-3 and Figure 23-3 describe the EFCPINS register.

Figure 23-3. EFC Pins Register (EFCPINS) [offset = 2Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
EFC Selftest Done	EFC Selftest Error	Reserved	EFC Single Bit Error	EFC Instruction Error	EFC Autoload Error	Reserved	
R-0	R-0	R-0	R-x	R-0	R-x	R-x	

7	6	5	4	3	2	1	0
Reserved							
R-x							

LEGEND: R = Read only; -*n* = value after power-on reset (nPORRST); x = Indeterminate

Table 23-4. EFC Pins Register (EFCPINS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15	EFC Selftest Done		This bit can be polled to determine when the EFC ECC selftest is complete
		0	EFC ECC selftest is not complete.
		1	EFC ECC selftest is complete.
14	EFC Selftest Error		This bit indicates the pass/fail status of the EFC ECC Selftest once the EFC Selftest Done bit (bit 15) is set.
		0	EFC ECC Selftest passed.
		1	EFC ECC Selftest failed.
13	Reserved	0	Reads return zeros. Do NOT write a 1 to this bit.
12	EFC Single Bit Error		This bit indicates if a single bit error was corrected by the ECC logic during the autoload after reset.
		0	No single bit error was detected.
		1	A single bit error was detected and corrected.
11	EFC Instruction Error		This bit indicates an error occurred during a factory test or program operation. This bit should not be set from normal use.
		0	No instruction error detected.
		1	An error occurred during a factory test or program operation.
10	EFC Autoload Error		This bit indicates that some non-correctable error occurred during the autoload sequence after reset. This bit also sets ESM group 3, channel 1.
		0	The autoload function completed successfully.
		1	There were non-correctable errors during the autoload sequence.
9-0	Reserved	0-1	After reset, these bits are indeterminate and reads return either a 1 or 0.

23.4.3 EFC Error Status Register (EFCERRSTAT)

Figure 23-4 and Table 23-5 describe the EFCERRSTAT register.

Figure 23-4. EFC Error Status Register (EFCERRSTAT) [offset = 3Ch]

31	8
Reserved	
R-0	

7	6	5	4	0
Reserved		Instruc Done	Error Code	
R-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after power-on reset (nPORRST)

Table 23-5. EFC Error Status Register (EFCERRSTAT) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved	0	Reads return zeros, writes have no effect.
5	Instruc Done		Instruction done. Used to indicate that the eFuse self test has completed
		0	The eFuse controller is still executing.
		1	The eFuse controller has completed executing.
4-0	Error Code		The error status of the last instruction executed by the eFuse Controller
		0	No error.
		5h	An uncorrectable (multibit) error was detected during the power-on autoload sequence.
		15h	At least one single bit error was detected and corrected during the power-on autoload sequence.
		18h	The signature generated by the ECC self-test logic did not match the golden signature written in the EFCSTSIG register. The EDAC circuitry might have a fault.
		All other values	All other values are reserved for e-fuse system tests and are not expected to occur in normal system use.

23.4.4 EFC Self Test Cycles Register (EFCSTCY)

Figure 23-5 and Table 23-6 describe the EFCSTCY register.

Figure 23-5. EFC Self Test Cycles Register (EFCSTCY) [offset = 48h]

31	16
Cycles	
R/W-0	

15	0
Cycles	
R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after power-on reset (nPORRST)

Table 23-6. EFC Self Test Cycles Register (EFCSTCY) Field Descriptions

Bit	Name	Description
31–0	Cycles	This register is used to determine the number of cycles to run the eFuse controller ECC logic self test. It is recommended to use a value of 600 (0x00000258).

23.4.5 EFC Self Test Signature Register (EFCSTSIG)

Figure 23-6 and Table 23-7 describe the EFCSTSIG register.

Figure 23-6. EFC Self Test Cycles Register (EFCSTSIG) [offset = 4Ch]

31	16
Signature	
R/W-0	

15	0
Signature	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after power-on reset (nPORRST)

Table 23-7. EFC Self Test Cycles Register (EFCSTSIG) Field Descriptions

Bit	Name	Description
31–0	Signature	This register is used to hold the expected signature for the eFuse ECC logic self test. It is recommended to write a value of 0x5362F97F to this register and a value of 600 (0x00000258) to the EFCSTCY register. If after running the eFuse ECC logic self test, the calculated signature does not match the expected signature in the EFCSTSIG register, then a value of 18h is stored in the EFCERRSTAT register.

Revision History

Changes from January 9, 2016 to February 28, 2018 **Page**

- **Chapter 1: Introduction** 57
- **Chapter 2: Architecture** 62
- Table 2-2: Changed table. Added Access Mode column 65
- Table 2-6: Corrected value in Valid RAM Groups column for number 13 73
- Table 2-6: Corrected value in Valid RINFO Register Value column for numbers 31 and 32 73
- Section 2.3.1: Updated third sentence in second paragraph. Changed to minimum of 32 peripheral clock (VCLK) cycles 76
- Table 2-28: Changed Description of bits for Value = 0 (Read) to enabled 94
- Section 2.5.1.13: Added second paragraph to NOTE 95
- Table 2-42: Corrected Description of PLLMUL bit. Value = 0h is ×1, Value = 100h is ×2 110
- Table 2-52: Changed Description of PLL1_FBSLIP_FILTER_ COUNT and PLL1_FBSLIP_FILTER_ KEY bits 122
- Section 2.5.1.44: Changed paragraph 129
- Section 2.5.1.44: Added NOTE 129
- Section 2.5.1.45: Changed NOTE 130
- Table 2-62: Added Note to VCLK2R and VCLKR bits 130
- Section 2.5.1.48: Added NOTE 132
- Figure 2-52: Changed Reserved bits to 2-0 133
- Figure 2-52: Deleted MPMODE bit 133
- Table 2-66: Changed Reserved bits to 2-0 133
- Table 2-66: Deleted MPMODE bit 133
- Table 2-74: Changed Description of PLL1_SLIP_FILTER_COUNT and PLL1_SLIP_FILTER_KEY bits 140
- Section 2.5.2.4: Changed paragraph 141
- Figure 2-61: Corrected register bit fields 141
- Table 2-76: Changed table to reflect updated register bit fields 141
- Section 2.5.2.5: Changed paragraph 142
- Figure 2-62: Corrected register bit fields 142
- Table 2-77: Changed table to reflect updated register bit fields 142
- Table 2-85: Corrected register names in Description of PROTSET bit for Value = 1 (Write) 147
- Table 2-86: Corrected register names in Description of PROTSET bit for Value = 1 (Write) 148
- Table 2-87: Corrected register names in Description of PROTSET bit for Value = 1 (Write) 148
- Table 2-88: Corrected register names in Description of PROTSET bit for Value = 1 (Write) 149
- Table 2-89: Corrected register names in Description of PROTCLR bit for Value = 1 (Write) 149
- Table 2-90: Corrected register names in Description of PROTCLR bit for Value = 1 (Write) 150
- Table 2-91: Corrected register names in Description of PROTCLR bit for Value = 1 (Write) 150
- Table 2-92: Corrected register names in Description of PROTCLR bit for Value = 1 (Write) 151
- **Chapter 3: I/O Multiplexing and Control Module (IOMM)** 159
- **Chapter 4: F021 Flash Module Controller** 173
- Section 4.1.2: Added definition for ATCM 174
- Section 4.6.2: Updated second paragraph 184
- Table 4-8: Updated Description of test modes 184
- Section 4.6.2.1: Updated eighth sentence in second paragraph. Changed ECC_MUL_ERR to ERR_PRF_FLG 185
- Section 4.6.2.5: Added NOTE 186
- Section 4.6.2.5: Updated second paragraph 186
- Section 4.6.2.6: Updated second sentence in second paragraph. Corrected the value. 187
- Section 4.6.2.6: Updated first sentence in fourth paragraph. Changed ECC_MUL_ERR to B1_UNC_ERR 187
- Section 4.6.2.6: Updated steps in fifth paragraph 187
- Table 4-9: Updated Name of test modes 187
- Table 4-9: Updated test mode 7. Changed ECC_MUL_ERR to B1_UNC_ERR 187
- Table 4-10: Updated Name of test modes 188

- Table 4-10: Updated test mode 1. Deleted D_MUL_ERR 188
- Table 4-10: Updated test mode 1. Changed ERR_MUL_FLG to ERR_PRF_FLG 188
- Table 4-10: Updated test mode 1. Deleted EE_D_MUL_ERR 188
- Table 4-10: Updated test mode 1. Changed EE_ERR_MUL_FLG to EE_ERR_PRF_FLG 188
- Table 4-10: Updated test mode 3. Changed EE_MAL_ERR to EE_CME 188
- Table 4-10: Updated test mode 4. Changed EE_COM_MAL_GOOD to EE_CMG and EE_MAL_ERR to EE_CME 188
- Table 4-11: Updated Name of test modes 189
- Table 4-14: Changed Description of EDACMODE bit 192
- Table 4-24: Changed Description of PROTL1DIS bit 204
- Section 4.7.22: Added paragraph 213
- Table 4-34: Changed Description of EMU_DMSW bit 213
- Section 4.7.23: Added paragraph 213
- Table 4-35: Changed Description of EMU_DLSW bit 213
- Section 4.7.24: Added paragraph 214
- Table 4-36: Changed Description of EMU_ECC bit 214
- Section 4.7.25: Added paragraph 215
- Table 4-37: Changed Description of EMU_ADDR bit 215
- Section 4.7.26: Deleted second paragraph 216
- Table 4-38: Updated Description of DIAG_MODE bit 216
- Section 4.7.30: Deleted paragraph 220
- Section 4.7.35: Changed paragraph 225
- Table 4-47: Changed Description of EE_EDACMODE bit 225
- Table 4-52: Changed Description of EE_CMG bit 229
- Table 4-52: Added Description for bit 4, EE_CME 229
- **Chapter 5: Tightly-Coupled RAM (TCRAM) Module** 232
- Table 5-2: Updated Description of EMU_TRACE_DIS bit. Tracing of read data to RAM Trace Port (RTP) module during emulation mode access 238
- **Chapter 6: Programmable Built-In Self-Test (PBIST) Module** 246
- Section 6.3.1: Changed step 6. Deleted ROM interface clock 250
- Section 6.5: Changed second paragraph (write 1h) 253
- Section 6.6.1: Changed step 5. Deleted ROM interface clock 265
- Section 6.6.1: Changed step 12. Deleted ROM clock 265
- Section 6.6.2: Changed step 5. Deleted ROM interface clock 266
- Section 6.6.2: Changed step 11. Deleted ROM clock 266
- **Chapter 7: CPU Self-Test Controller (STC) Module** 267
- **Chapter 8: CPU Compare Module for Cortex-R4 (CCM-R4)** 283
- **Chapter 9: Oscillator, PLL, and Clock Monitoring** 291
- Section 9.5: Updated third paragraph. Changed $f_{(HCLK)}$ to $f_{(GCLK)}$ 300
- Table 9-1: Updated Frequency Limit value to $f_{(GCLK)}$ for $f_{PLL\ CLK}$ 300
- Table 9-2: Updated formula for NF 301
- Section 9.5.4: Added last sentence to step 3 307
- Section 9.8: Changed step 3, second sentence 315
- **Chapter 10: Dual-Clock Comparator (DCC) Module** 317
- Table 10-13: Changed Description of KEY bit for Value = All other values 330
- **Chapter 11: Error Signaling Module (ESM)** 331
- Section 11.1.2: Changed first paragraph 332
- **Chapter 12: Real-Time Interrupt (RTI) Module** 353
- Equation 24: Corrected first eqution (if RTICPUCy ≠ 0) 357
- Figure 12-13: Updated Read/Write value of bits to R/WP-0 367
- Figure 12-13: Updated LEGEND to include WP 367
- **Chapter 13: Cyclic Redundancy Check (CRC) Controller Module** 390
- **Chapter 14: Vectored Interrupt Manager (VIM) Module** 403
- Section 14.4: Added NOTE 411
- **Chapter 15: Enhanced Quadrature Encoder Pulse (eQEP) Module** 433

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265